Matrix Transform Imager Architecture for On–Chip Low–Power Image Processing

A Thesis
Presented to
The Academic Faculty

by

Abhishek Bandyopadhyay

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
July 2004

Matrix Transform Imager Architecture for On–Chip Low–Power Image Processing

Approved by:

Professor Paul Hasler, Adviser

Professor Joel Jackson

Professor David Anderson

Dr. Mark T. Smith
(Hewlett-Packard Laboratories)

Professor Stephen DeWeerth

Date Approved: 27 July 2004

To my parents

ACKNOWLEDGEMENTS

I wish to gratefully acknowledge my advisor, Dr. Paul Hasler, for helping me during my stay at Gatech for three years, providing an opportunity, guiding my research and reviewing this thesis. I also want to thank Dr. David Anderson, and Dr. Joel Jackson for all the discussions we had and for reviewing this thesis. Many thanks also to the committee for the fruitful reviews.

I would like to thank Jungwon Lee for all the help he provided and the fruitful discussions we had during my stay at Gatech. I had a very enjoyable time working with all the members of icelab specially David Abramson, Faik Baskaya, Ravi Chawla, Ryan Robucci, Guillermo Serrano, and Venkatesh Srinivasan.

Many thanks are due to Haw–Jing Low and Heejong Yoo for proof reading the thesis.

Last but not the least, I want to thank all my friends for all their help and for the wonderful time I had during my stay in Atlanta.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Camera–on–a–chip systems have tried to include carefully chosen signal processing units for better functionality, performance and also to broaden the applications they can be used for. Image processing sensors have been possible due advances in CMOS active pixel sensors (APS) and neuromorphic focal plane imagers. Some of the advantages of these systems are compact size, high speed and parallelism, low power dissipation, and dense system integration. One can envision using these chips for portable and inexpensive video cameras on hand–held devices like personal digital assistants (PDA) or cell–phones

In neuromorphic modeling of the retina it would be very nice to have processing capabilities at the focal plane while retaining the density of typical APS imager designs. Unfortunately, these two goals have been mostly incompatible. We introduce our MAtrix Transform Imager Architecture (MATIA) that uses analog floating–gate devices to make it possible to have computational imagers with high pixel densities. The core imager performs computations at the pixel plane, but still has a fill–factor of 46 percent—comparable to the high fill–factors of APS imagers. The processing is performed continuously on the image via programmable matrix operations that can operate on the entire image or blocks within the image.

The resulting data-flow architecture can directly perform all kinds of block matrix image transforms. Since the imager operates in the subthreshold region and thus has a low power consumption, this architecture can be used as a low–power front end for any system that utilizes these computations. Various compression algorithms (e.g. JPEG), that use block matrix transforms, can be implemented using this architecture. Since MATIA can be used for gradient computations, cheap image tracking devices can be implemented using this architecture. Other applications of this architecture can range from stand-alone universal transform imager systems to systems that can compute stereoscopic depth.

CHAPTER I

EXISTING APPROACHES IN VISION PROCESSING

1.1 Overview of Human visual system

Light is the electromagnetic radiation that stimulates our visual response.Visual perception occurs in two stages: light being converted into electrical signals by a special sensory organ called the retina, and transmission of these signals through the optical nerve to the brain for further processing of these signals. The retina of the human eye contains two types of photoreceptors called the rods and cones. Figure 1 shows the cross–sectional view of a human eye. The rods are about 100 million in number and provide *scotopic* vision (visual response at low magnitudes of illumination). The cones are fewer in number, have less sensitivity compared to rods and provide *photopic* vision. Cones mediate colour vision and provide greater spatial and temporal resolution. Unlike most neurons, rods and cones do not fire action potential instead they respond to light with graded changes in membrane potential [66]. This change in membrane potential is caused by change in ionic fluxes across the membranes of the rods and cones.

The output neurons of the retina are the *ganglion cells.* Their axons form the optic nerve which project to the Lateral Geniculate Nucleus (LGN) and other parts of the brain. Unlike the photoreceptors the ganglion cells transmit information as trains of action potential. In between the ganglion cells and the photoreceptors are three classes of interneurons: *bipolar, horizontal,* and *amacrine* cells (Fig. 2). These cells transmit signals from the photoreceptor to the ganglion cells and also combine signals from several photoreceptors, so that the electrical responses evoked in ganglion cells depend critically on the image pattern and the temporal changes associated with the image. Cone signals are conveyed to the ganglion cells through direct and lateral pathways. The properties of the ganglion cells enhance the ability of detecting weak contrasts and rapid changes in visual image. Under low light conditions ganglion cells cease to be detectors of local contrast and instead become effective

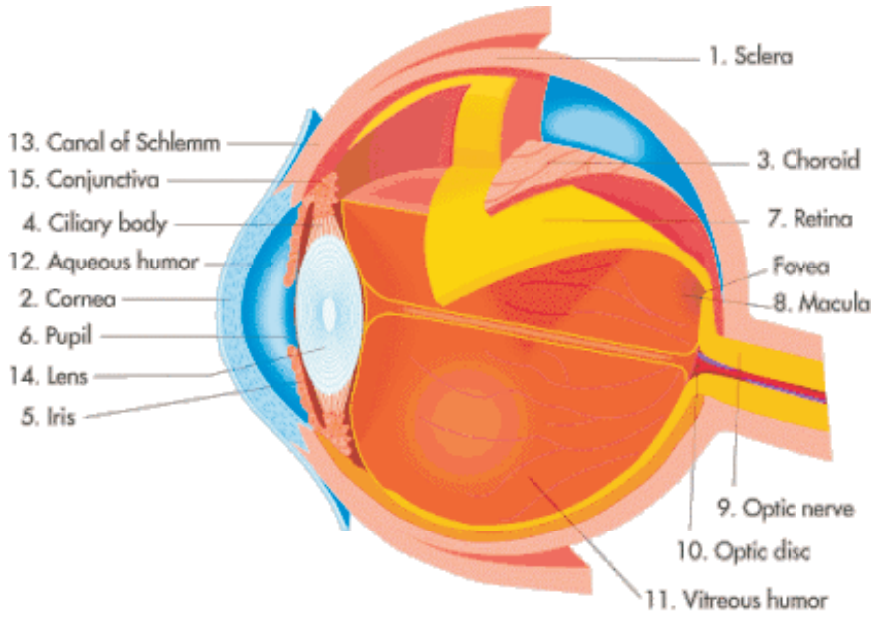


Figure 1: Human Eye: Cross–sectional view of the Human eye. The retina of the human eye contains two types of photoreceptors called the rods and cones.

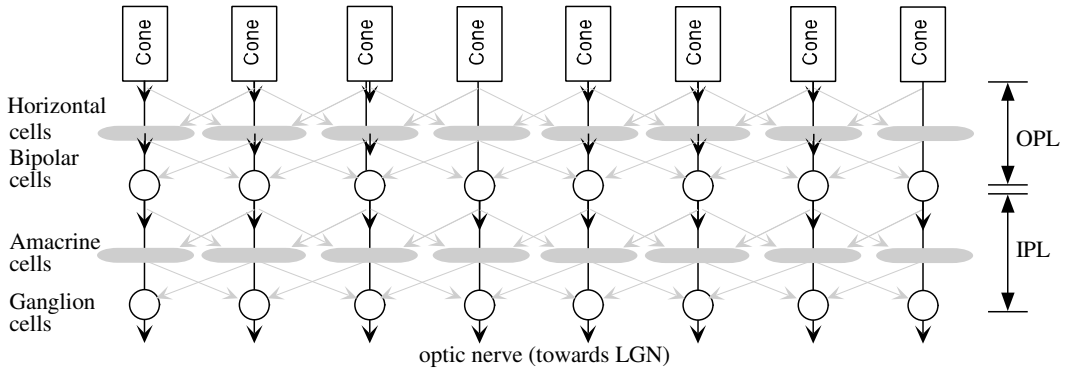


Figure 2: Classic diagram of vertebrate retina: Retina is organized as layers of neurons with various spatial interactions; these layers are organized as the Outer Plexiform Layer (OPL) and Inner Plexiform Layer (IPL). Gap junctions form electrical synapses (as opposed to typical chemical synapses) between neurons on a given layer. The retina model implements various linear and non-linear spatio-temporal filters whose outputs are further filtered through visual cortex.

light detectors. The organizational principles of the visual system are the segregation of information into parallel processing pathways and the shaping of response by inhibitory lateral connections.

1.2 Photodiode and photoconduction

The fundamental purpose of photo detectors is to convert an input photonic signal into an electrical signal. The application in which the detector is used affects the performance criterion. For imaging applications a photodetector should provide good spatial resolution, gray-scale resolution, need to be two dimensional without introducing spatial aliasing in either direction, should be able to operate at both low and high light intensities, etc.

Inner photo electric effect is the process by which an electron absorbs incident photonic energy. If this energy is more than the bandgap energy then electrons from the valence band are excited to the conduction band. Hence in the exposed area illumination increases the concentration of mobile charge carriers above the thermal equilibrium value. *Photoconduction* happens when separation of electron hole pairs, under the influence of an electric field, contribute to an electric output signal. A diode is a photosensor because the presence of depletion region reduces dark current, while the built-in potential enables charge separation even in the absence of externally applied fields. The resultant reverse bias current is the photocurrent. Figure 3 shows the principle of operation of a photodiode. Generation of excess carriers within the depletion region enhances the reverse bias current. Therefore, if a reversed biased junction is illuminated with light energy in excess of the bandgap energy, the reverse bias current is significantly increased because of the presence of photogenerated carriers. The reverse bias also aids the operation of the device. This bias enhances absorption by increasing the depletion width in which the light is absorbed. The photogenerated electron-hole pairs are separated by the action of the drift field. Hence, if the reversed bias is sufficiently high high carrier drift velocities are obtained and impact ionization can occur. This implies that carrier multiplication and current gain can occur in the device.

Photodiodes provide high quantum efficiency and high bandwidths. The bandwidth depends on the drift time within the depletion region, the capacitance of the depletion region, and the carrier diffusion to the depleted region [17]. Increasing the depletion width increases the incident radiation absorption but reduces the bandwidth by increasing the transit time across the junction. The speed also depends on the junction capacitance.

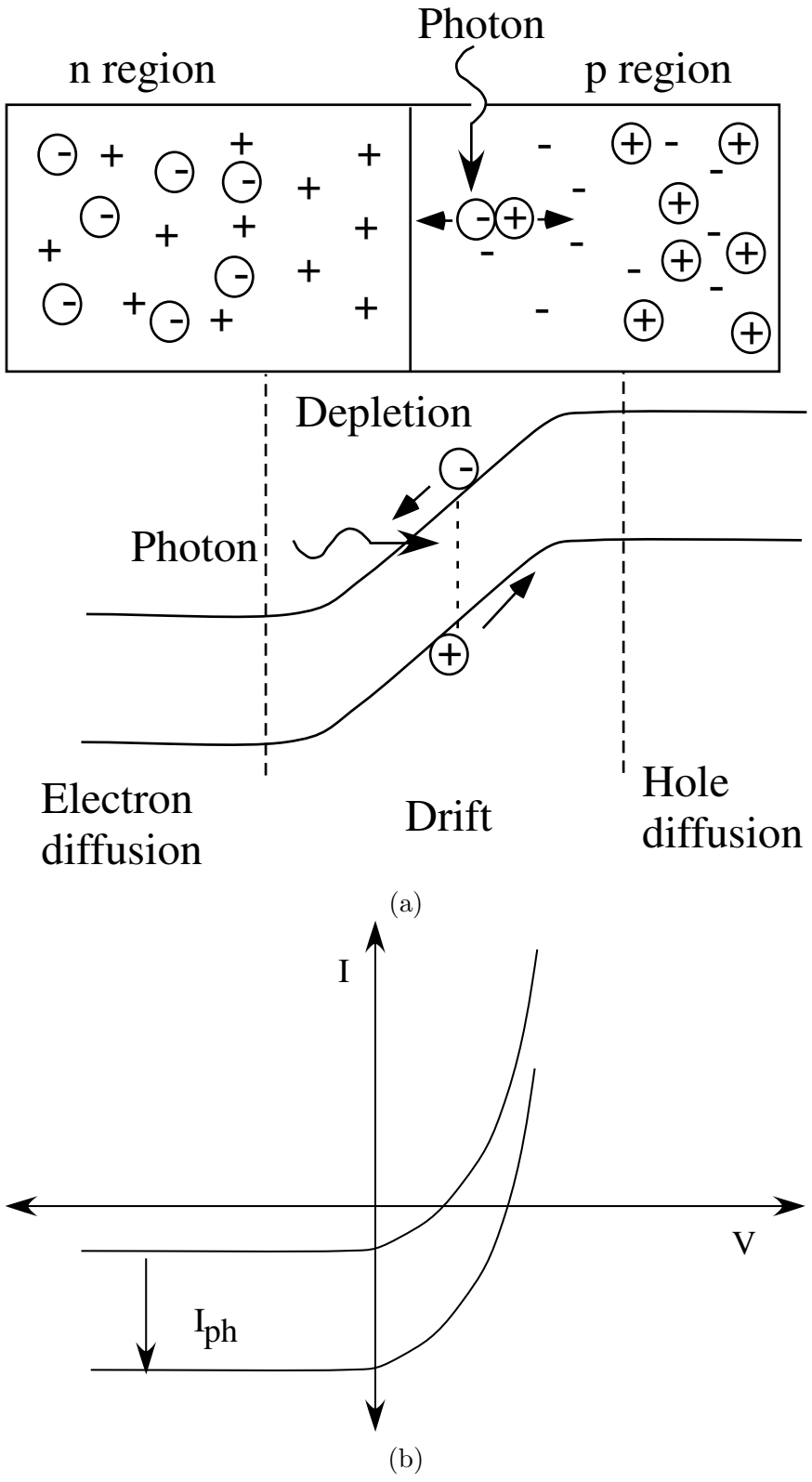


Figure 3: Principle of photo diodes: (a)Schematic showing the effect of incident photons and the creation of electron–hole pairs. (b)Current–to–voltage characteristics of a photodiode. It has the same shape as a normal diode. Illumination causes the curve to shift down, as shown in the figure.

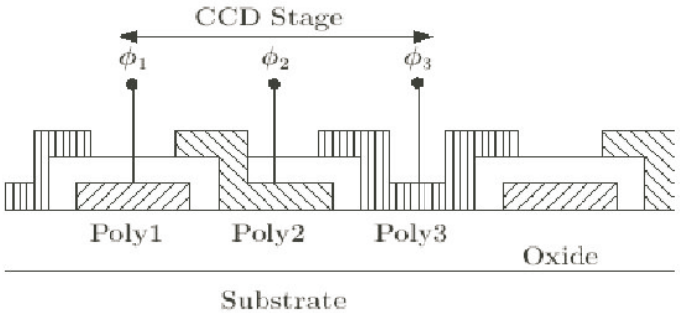


Figure 4: Cross–section of a CCD: Cross–sectional view of a charge coupled device. It shows the three polysilicon gates.

1.3 Charge coupled device

The charge coupled device (CCD) has dominated the digital imaging market due to it's low fixed pattern noise (FPN), low readout noise, higher quantum efficiency and higher sensitivity. A CCD consists of an array of photogates that integrate the photogenerated charge [17]. The charge integration of the whole array takes place at the same time. The signal charge packets of the different pixels are simultaneously moved underneath the semiconductor surface towards the readout circuitry. Most CCDs are either *buried channel* CCDs that consists of bulk-storage photogates, or *surface channel* CCDs where storage and transport occurs at the semiconductor interface. Buried–channel CCDs are fabricated using special processes and suffer from blooming - spilling of excess charge into adjacent packets when charge packet gets overfilled. Surface–channel CCDs are fabricated on CMOS–compatible processes, but suffers from inefficient charge transfer across a large array, as the silicone-oxide interface of a standard CMOS process does not guarantee efficient charge transfer, as required for CCD imaging.

A single CCD pixel (picture element) is illustrated in Fig. 4. This figure shows three polysilicon gates oriented perpendicular to two-channel stop regions. Between the channel stop regions lies the buried channel. If the potential on the middle electrode is more positive than that applied to either of the other two gates, a local potential energy minimum will be formed under the middle gate. When photons strike the pixel, electron-hole pairs are created via the photoelectric effect. Electrons created within the potential minimum will

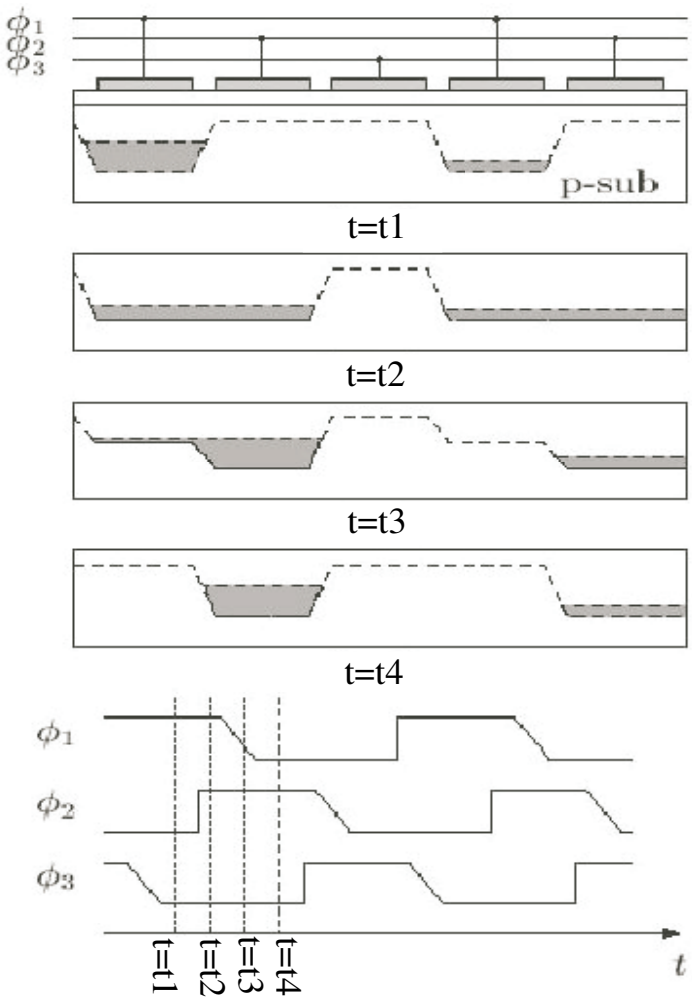


Figure 5: Clocking for CCD: Schematic of charge transport using a three phase clocking scheme. The charge packets, symbolized by the hatched areas, are shifted from left to right as time progresses.

be collected there. Electrons that are created in the channel stop region or in the substrate beneath the pixel may diffuse to the minimum potential area and be collected. In either case, the holes diffuse to and are collected in the p-type substrate. The quantity of charge that is collected in the well is linearly related to the intensity of the photon flux and to the time over which the light is allowed to fall on the pixel (integration time). The prevalence of the three phase technology is principally due to high yield and process tolerance of this technology. A single CCD pixel has no means to read out the quantity of charge accumulated beneath the integrating electrode. The process of reading out this signal charge involves moving the packet from the site of collection to a charge detecting amplifier located at the

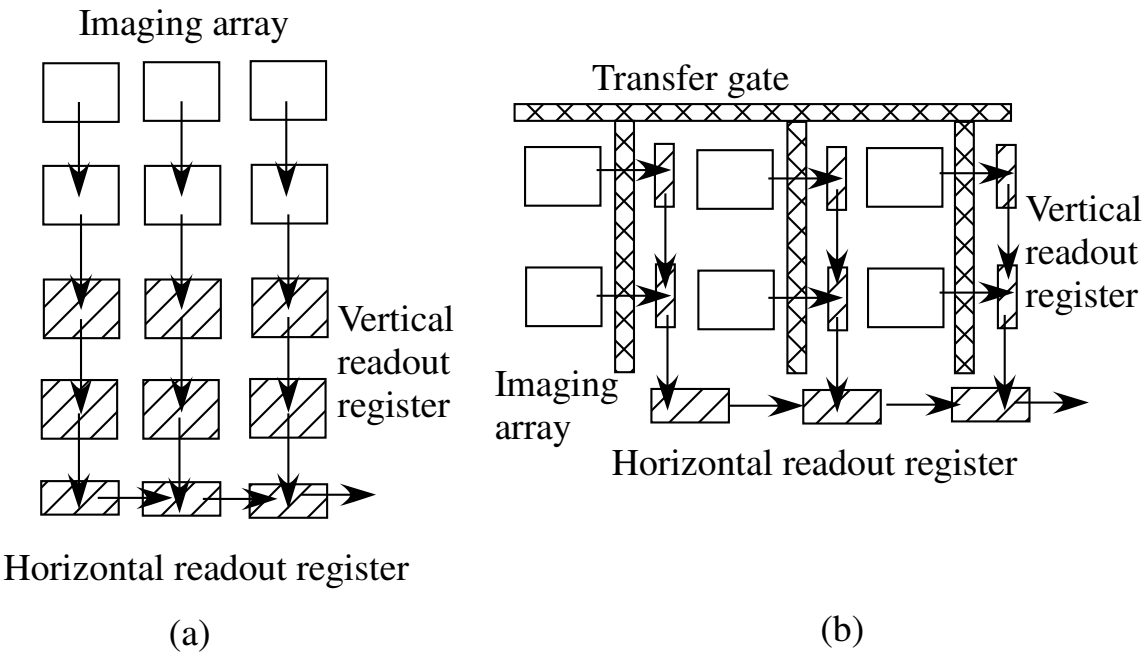


Figure 6: CCD architectures: (a) Frame–transfer CCD and (b) interline–transfer. The hatched areas indicate MIS gates that are shielded from light. The arrows indicate the direction of charge flow [80].

end of the linear array. Figure 5 shows the three phase clocking scheme required to read charge information from a CCD imager.

Quantum efficiency (QE) is the measure of the efficiency with which incident photons are detected. Some incident photons may not be absorbed due to reflection or may be absorbed where the electrons cannot be collected. The quantum efficiency is the ratio of the number of detected electrons divided by the product of the number of incident photons times the number of electrons each photon can be expected to generate [17].

CCDs can be categorized according the manner in which the charge is readout. *Frame–transfer* CCDs (Fig. 6(a))transport the charge underneath the photogates themselves, while *Interline–transfer* CCDs first transfer charge packets to parallel running lines of MIS gates and the charge is then transported underneath those (Fig. 6(b)) [80]. Frame–transfer CCDs have higher fill–factor but Interline–transfer CCDs have a simpler clocking scheme.

CCDs have various disadvantages:

- Large number of clock signals are required to operate a CCD system. These are relatively high voltage clocks and hence the system has a high power budget.

- Since CCDs are tuned for charge transfer quality the MOS transistors available in these processes have poor characteristics.

- Multi–die system or a multi–chip system is required for performing complex image processing. Addition and subtraction are the only operations available in a CCD process. Hence system integration is not easy when CCDs are used for image capture.

1.4 CMOS imagers

In most machine vision systems the image capture and processing modules are viewed as two separate systems, and often no interaction between these stages exist. On the other hand the biological approach towards computer vision attempts to benefit by emulating biological systems based on basic observations of these systems. The two features that make the biological system very efficient are the relative simplicity of the visual processing elements and the parallel computation that takes place in the visual pathway. Biological systems have close interaction between the retina (image capture) and the subsequent opto-neuro pathways (processing). Vision chips that mimic this interaction by integrating both of these stages on the same chip offer several advantages over the conventional systems, including compact size, higher speed and parallelism, power dissipation and dense system integration.

CMOS technology provides an attractive alternative to CCD technology or implementing low-power, low-cost images with high levels of integration. In CMOS imagers we see a tradeoff between large–scale focal–plane processing, and fill factor. In neuromorphic imagers most of the computation is performed at the pixel-level and thus the fill–factor is small. On the other hand passive pixel sensors (PPS) and active pixel sensors (APS) use the pixel to read the photo-transduced current and most of the processing is performed at the periphery, hence the fill factor in these systems are high.

1.4.1 Passive Pixel Sensor

Passive pixel sensor (PPS) were the first image-sensor devices used in the 1967 [112]. In passive-pixel CMOS sensors, as shown in Fig. 7, a photodiode converts photons into an

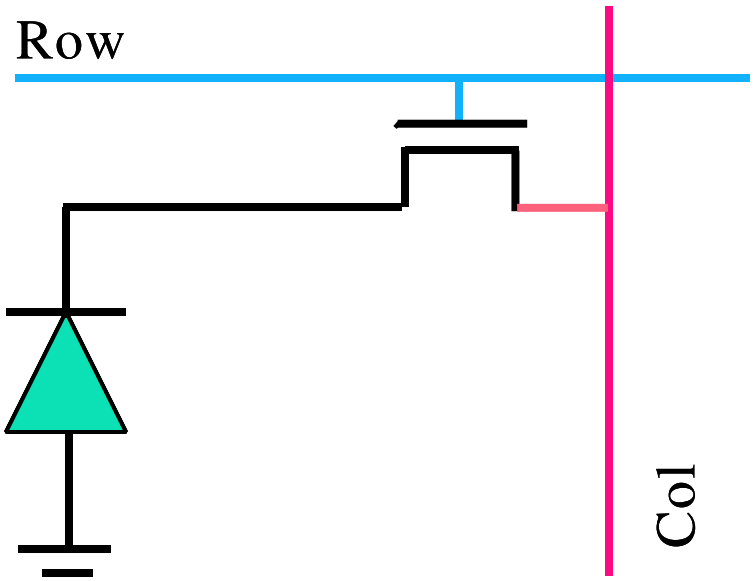


Figure 7: Passive pixel sensor: Schematic of a passive pixel sensor (PPS). The readout in PPS architecture is destructive.

electrical charge. When the access transistor is activated, the photodiode is connected to a vertical column bus. A charge integrating amplifier readout circuit at the bottom of the column bus keeps the voltage on the column bus constant and reduces kTC noise [38]. When the photodiode is accessed, the voltage on the photodiode is reset to the column bus voltage, and the charge, proportional to the photo signal, is converted to a voltage by the amplifier. The single-transistor photodiode passive pixel allows the highest design fill factor for a given pixel size or the smallest pixel size for a given design fill factor for a particular CMOS process [38, 39].

The advantages are high fill–factor, true X-Y addressing possible by adding a second selective transistor, and high quantum efficiency, due to large fill–factor. Some of the disadvantages are that the readout noise level appears as a background pattern in the image and the fact that these imagers do not scale well to larger size and or/faster pixel readout rates. This is because both increased bus capacitance and faster readout speeds result in higher readout noise.

Though not many groups are using this technology there is still some research going on in this area. Fujimori has presented 256x256 CMOS differential passive pixel imager

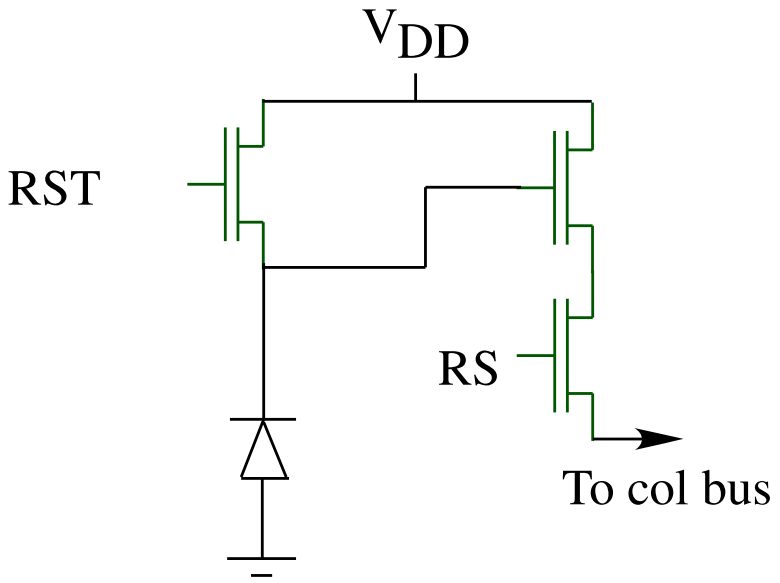


Figure 8: Active pixel sensor: Schematic of an active pixel sensor (APS). The amplifier is usually a source follower.

with FPN reduction techniques at ISSCC 2000 [41]. This chip was deigned in 0.6μm 2–poly 3–metal CMOS process and the pixel size was 20μm x 20μm with a fill–factor of 40%.

1.4.2 Active Pixel Sensor (APS)

These approaches, typically credited to Fossum, et al., [37, 38, 78, 114, 115] worked with photodiode–based arrays with minimal circuitry in the pixel (shown in Fig. 8), resulting in large imaging arrays, and therefore, a technology viable for digital cameras and more sophisticated peripheral computations. A sensor with an active amplifier within each pixel is referred to as an active pixel sensor or APS. Since each amplifier is only activated during readout, power dissipation is minimal and is generally less than a CCD. Pixels are typically designed for a fill factor of 20–30%. Some of the recent pixels are 3μmx3μm or 4μmx4μm in size. Loss in optical signal is more than compensated by reduction in read noise for a net increase in signal-to-noise (S/N) ratio and dynamic range.

Much work has been done in improving the quality of pictures from an APS imager [24, 36, 39, 50, 22]. These imagers usually contain a APS array, readout circuitry, on–chip ADC for fast readouts. There have been many implementations where peripheral circuitry is added for post processing on images. Some of the recent work in APS imagers are discussed

below.

Cho has presented a 1.5–V 550-μW 176x144 autonomous CMOS APS sensor [21, 22]; Krymski has reported a high-speed, 240-frames/s, 4.1-Mpixel CMOS sensor [74]; Takayanagi has reported a 1 1/4 inch 8.3M pixel digital output CMOS APS for UDTV application [106].

Sodini's group has been actively working on APS imagers [26, 45]. Recently they have published a predictive multiple sampling algorithm with overlapping integration intervals for linear wide dynamic range integrating image sensors [2] and a 256 x 256 CMOS imaging array with wide dynamic range pixels and column–parallel digital output [26].

Gamal's group has also done extensive studies of APS imagers. Discussion on temporal noise in CMOS photodiode can be found in [109]. His group has also looked into designing ADCs for fast image readout [116, 117]

1.4.3 Neuromorphic imagers

Neuromorphic imagers are essentially vision chips that try to mimic general characteristics of the vertebrate retina like adaptation to light intensity and edge enhancement. Early retina model systems used focal–plane processing to mimic the edge enhancement properties in the early retina processing based on photodiodes and phototransistors that naturally occur in a silicon CMOS process [5, 84, 87]. Another group of spatial signal processing chips target more global features like object position, orientation, and centroid. In foveated sensors, the physical size and placement of the photoreceptors form a linear–polar or log–polar mapping on the image. Neuromorphic imagers chips are characterized by significant amounts of signal processing occurring at the image plane, but usually at the cost of a small fill–factor. Retinal processing imagers and research are focused primarily on machine vision tasks where the required pixel count can be smaller thus emulating lower order biological systems; for example, flies can accomplish amazing things with a small number of pixels [51, 52]. Although much can be explored in vision problems at the level of flies, many neuromorphic visual signal processing systems aim toward modelling much larger organisms. Later designs improved so as to be usable in systems at high density levels [16, 5, 14] and for high performance [28]. From these retina chips, several higher level processing ICs have

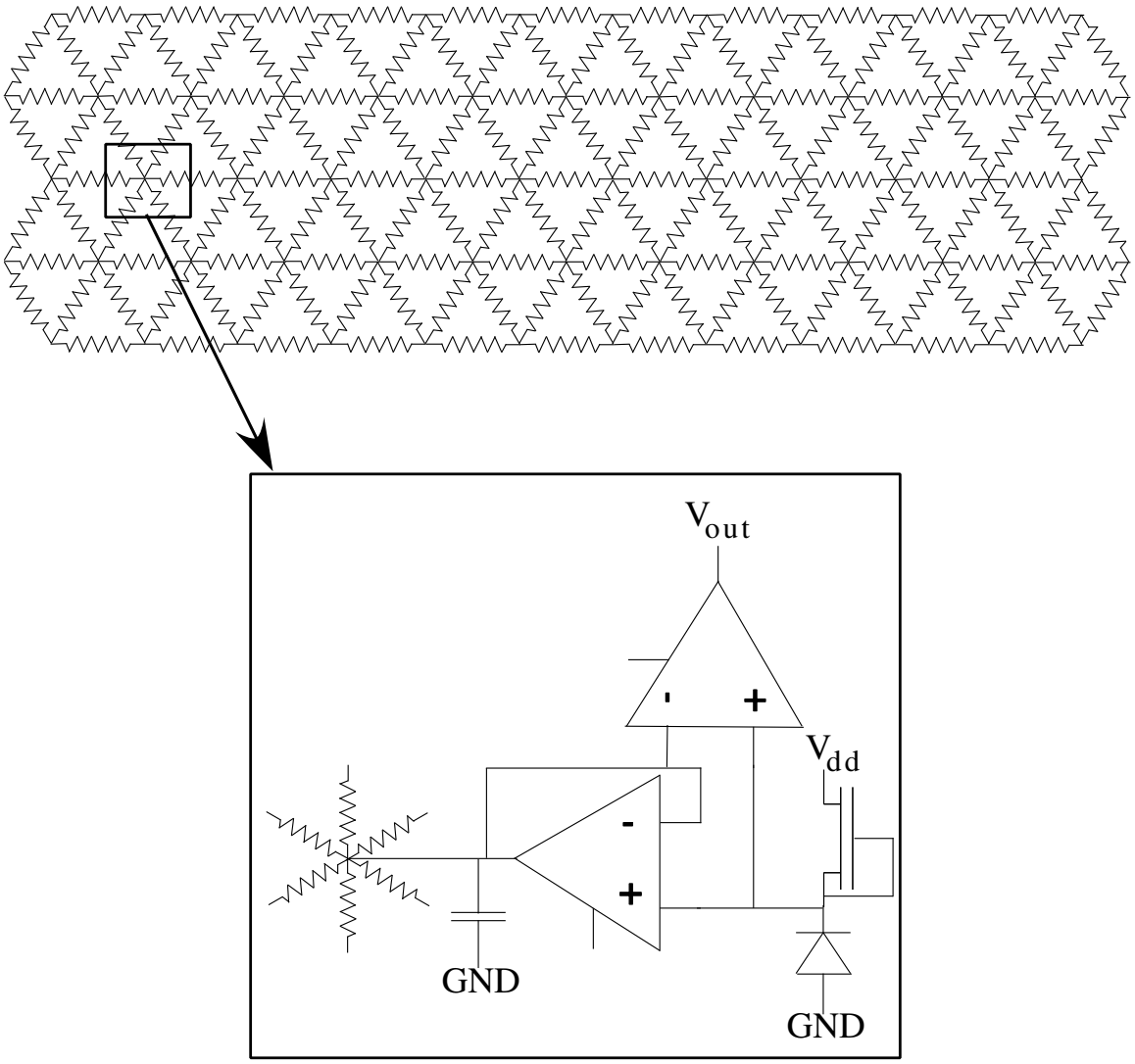


Figure 9: **Silicon retina**: Circuit diagram of the first retina implementation by Mahwold and Mead [84]. The spatial distribution of resistors is used to perform spatial filtering which models the gap junctions and some of the computation of the horizontal cells in IPL. The inset shows the basic circuit element at each node. Each node has a photoreceptor circuit built from an on-chip photodiode and log-compression of the resulting signal. The gm-C filter is used to create a temporally high-pass, spatially edge-enhanced, filtered output.

been built to investigate stereo processing [83, 82], communication architectures for action potentials [15], attention computations, and motion [29, 30, 51, 52, 64, 100, 108]. Typically, because of the large pixel size associated with the large number of transistors in each pixel, image sensors with retinal computation typically only have a fairly small number of pixels on a given IC. In only a very few cases will one see more than 50k image elements on a fairly large IC [5].

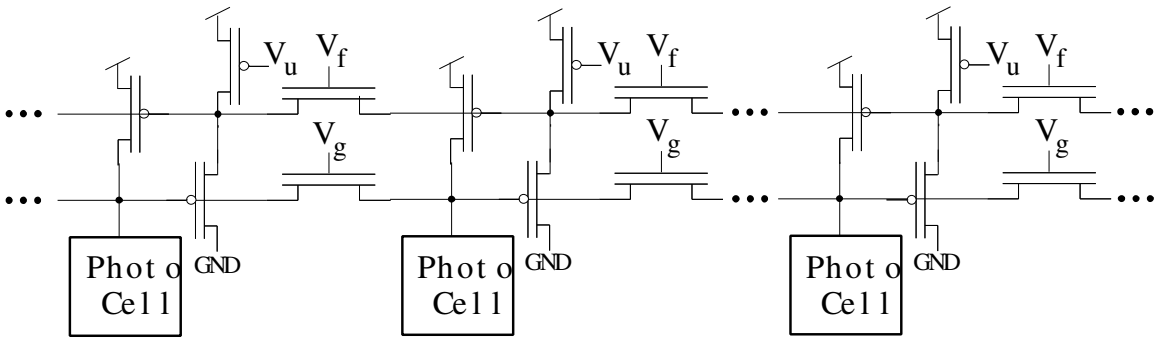


Figure 10: Implementation of OPL layer: Current mode implementation by Bohean and Andreou [16] of the OPL layer. This efficient implementation models many of the linear and nonlinear processing of this approach. Since this implementation, there have been several additional models for the entire retina based upon these techniques.

Mahowald's silicon retina was among the first neuromorphic vision chips [84, 87]. This chip performed computation based on the distal layers of the retina. The cones were the light detectors. These were implemented using phototransistors and diode connected I-Vs. Bipolar cells detect the difference between the input and the averaged output of the horizontal cells. Averaging is performed using a hexagonal network of active resistors, as shown in Fig. 9 . An exponential smoothing operation is performed by the resistive network. The smoothing factor depends on the diffusion constant in semiconductors (value of the resistors). An enhanced version of this silicon retina was Mead's adaptive retina. This chip used floating–gate MOSFET (FGMOS) as a feedback element for offset and mismatch correction [86, 88].

The outer–plexiform of retinal processing layer has been modelled on silicon by Andreou and Boahen [16]. The implementation uses a compact circuit, which enabled the realization of a 210x230 array of image sensors. A diffusive network, as shown in Fig. 10, is used in this chip. This circuit includes two layers of the diffusive networks. The upper layer corresponds to the horizontal layer, while the lower layer to cones.

Blum *et.al.* have designed a CMOS imager with real–time frame differencing and centroid computation [13, 31]. The current steering circuitry in the pixel allows for direct readout, programming, x–centroid computation and y–centroid. The pixel size is $900\mu m^2$ with a fill–factor of 18%. The centroid computation is based on the DeWeerth's aggregation

network. A spatially sweeping reference voltage is required for this circuit. This is provided by a resistive voltage divider with it's ends connected to reference voltages. Polysilicon resistors are used for the voltage divider. An analysis for constant background illumination and constant width and intensity is given in [31].

In Mitshubishi's CMOS retina chip the photocurrent output from a photodiode is directly modulated [43, 44]. It is an artificial retina chip consisting of an array of variable sensitivity photo–detector cells (VSPD) which accomplishes both nondestructive output and programmable positive or negative sensitivity. Despite its simple structure, this VSPD array realizes programmable focal plane image processing by employing between-pixel current mode calculations, together with a novel addressing method in which filtering is executed by varying the addressing pattern generated by a scanning unit. The chip consists of a 256x256 n-MOS array. The pixel size was 35μmx26μm with a fill–factor of 25% in a 2μm n-MOS process. The work has demonstrated image capture in video mode and edge extraction mode, and light spot tracing using pattern matching in conjunction with the random access function.

Tanner and Mead have reported a optic flow motion detector chip [108].The chip consists of phototransistors, and spatial and temporal differential circuits for computing the various gradients required for the algorithm. the computations are carried out collectively across the chip. The output of the chip is a global signal indicating the motion flow of the whole image. This is one of the first vision chips that demonstrated that mathematical algorithms can be implemented on VLSI.

Delbrück has presented a correlation–based motion detector [29]. This is a more robust method for motion detection as it uses correlation to extract information rather than gradients. The chip has an array of photodetectors, time delay elements, and anti–bump circuits. Delbrück has implemented a hexagonal 2–D motion sensor. Photodetection is performed using the novel adaptive photodetector pixel deigned by Delbrück [28], as shown in Fig. 11. The output of the motion detector is obtained from the anti–bump circuit.

Meitzler *et.al.* has described a 1–D motion detector chip similar to the Reichardt correlation based motion detectors [90]. In this implementation sample and hold circuits have

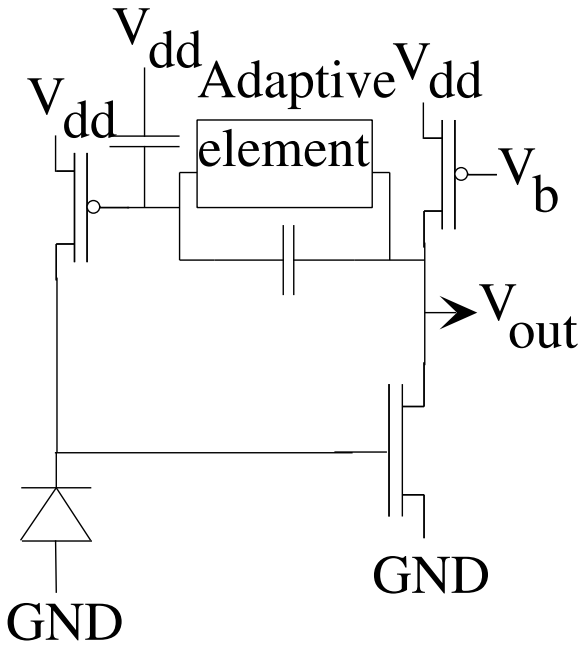


Figure 11: Adaptive photoreceptor: This circuit practically senses currents over seven orders of magnitude in light intensity (bright sunlight to candlelight).

been used instead of the delay elements of the Reichardt model. The front end of the chip uses Andreou retina cells. The sample and hold circuit has a relatively long retention time. In this chip, absolute difference function has been used to replace the multiplier units of a the Reichardt model. The chip was fabricated in 2μm CMOS process. Each of the 22 pixels occupied an area of 365x77μm^2.

Etienne–Cummings' has presented a motion detection chip based on a modified Reichardt algorithm, called temporal domain optical flow measurement (TDOFM) [35]. In this algorithm motion is detected by locating the zero–crossings and determining their appearance or reappearance. Velocity is calculated by computing the time an edge takes to move from one pixel to a neighboring pixel. A passive network is used to perform spatial smoothing of the input image.The chip consists of photodetectors, zero–crossing edge detectors, positive and negative edge motion pulse generators, correlator and speed measurement circuits. This chip was fabricated using a 2μm CMOS process.

1.5 Recent work on on–chip image transforms, convolutions and compression implementations

Recent progress in CMOS based image sensors and cameras has created new opportunities for developing low–power, low–cost, system–on–a chip cameras that include various image processing functions. Some to the recent work relevant to this thesis are discussed below:

- A four image reorganization IC for real–time difference encoding for hierarchial lossless image compression was reported by Kemeny *et.al.* Two image reorganization processors are realized on the focal plane and two were designed for hybridization to a separate IC. This represents the first integration of a 256x256 buried–channel frame–transfer CCD image sensor with additional charge domain circuits for video rate image reformatting. The reformatting circuitry occupied 2% of the chip area. Pixel data reorganization is performed through simultaneous readout of three rows of data, followed by pixel resequencing and sampling to provide differential output. This system of ICs provide real–time image reorganization to enable pyramidal, differential output of image data, thus simplifying downstream electronics, reducing power, size and weight of lossless hierarchial compression hardware. The drawback of this system is that it is a multi–chip system and hence interfacing is not trivial. Similar big pixel processors have been proposed by Paillet *et.al.* and Dudek *et.al.* Paillet *et.al* have designed a pixel consisting of 50 transistors containing a tiny digital processor [93]. In order to reduce the chip area required, the chip has a small grayscale resolution only. It is mainly suited for image segmentation and other related image processing like pattern recognition. Dudek and Hicks have presented a smart–sensor VLSI circuit for focal plane processing of gray–scale images. Each pixel has a processing unit consisting of a switched current microprocessor, and the architecture is based on a fine–grain software programmable SIMD array. A 21x21 prototype was fabricated in a 0.6μm CMOS process. The pixel size was 98.6μm x 98.6μm. Although the imager has flexible functionality the pixel size is large.

- Chiang and LaFranchise has proposed a programmable image processor that can be

used as a 2-D matched filter to perform spatial filtering with 20 different 7x7 spatial filters. Hence it can be used for applications such as noise removal, image enhancement and feature extraction. The processor was developed based on CCD technology. This chip was fabricated with a double–polysilicon, double metal, buried channel CCD/CMOS process.

- Spirig *et.al.* have designed a fully parallel focal plane CCD array which performs image acquisition and convolution with arbitrary kernels [105]. The real-time programmable spatial convolution is generated for all pixels in parallel during the exposure period. The 2-D convolution is performed by shifting a charge pattern in two dimensions and the exposure time is varied in proportion to the weight of each kernel coefficient. The size of the pixel is 63μm x 65μm. This CCD imager with convolution capabilities is suitable only for slow moving or static images.

- Luo and Harris have reported a on–sensor wavelet compressor for CMOS imager [81]. The Haar transform algorithm has been embedded into the architecture. An efficient transform scheme is proposed which required two computational units: a 2x2 element two–dimensional transform unit and an adder or subtractor. the image is readout in a column parallel fashion. The chip was designed and fabricated on a 0.5μm CMOS process. This imager contains a standard APS pixel. At the bottom of each column a network of switches and capacitors are used to perform the transform. CDS is used to suppress fixed pattern noise, and pixel KTC noise. This architecture is not programmable and can only perform 4x4 modified haar transform.

- A computational image sensor using conditional replenishment has been reported by Aizawa *et.al.* Conditional replenishment is employed for the compression algorithm [3]. Conditional replenishment is based on detection and coding of the moving areas so that it makes use of temporal redundancy to compress image signals. Conditional replenishment was originally proposed in the early age of image compression as a digital compression method which could be easily implemented in silicon. In each pixel, current pixel value is compared to that in the last replenished frame. The value

and the address of the pixel are extracted and coded if the magnitude of the difference is greater than a threshold. Analog circuits have been designed for processing in each pixel. The pixel was 190μm x 190μm in a 2μm CMOS process. Since the pixel size is large this is not suited for high resolution sensor.

- Sjöström *et.al.* had implemented a DCT chip for real–time video applications [101]. This chip used features like distributed arithmetic and application specific pipelined RAM memory. The implementation was not based on any "fast algorithm" but on a direct implementation of equations. Distributed arithmetic was used to compute the required inner products. This chip can be the sub–block of a bigger system also containing a pixel array. One or two dimensional transforms of lengths 4, 8, 16 can be calculated. Pipelining was also used in this architecture. The chip was designed using 2μm CMOS process.

- Etienne–Cummings *et.al.* have presented a programmable focal–plane Multiple Instruction Multiple Data (MIMD) image processor chip. The chip employs a MIMD architecture to provide five spatially processed images in parallel [36]. Gruev and Etienne–Cummings have designed a pseudogeneral image processor for realizing steerable spatial and temporal filters at the focal-plane based on previous work [50]. Figure 12 shows the schematic of the proposed architecture. The convolution of the image with programmable kernels is realized with area-efficient and real-time circuits. The chips architecture allows photoreceptor cells to be small and densely packed by performing all analog computations on the read-out, away from the array. The orientation of the spatial filters can be changed on the fly. The main components of the GIP are the 16 rows by 16 columns photo pixel array, three vertical and three horizontal scanning registers, and a single processing unit. The processing unit, which consists of four identical but independent sub–processors, is implemented with digitally controlled analog multipliers and adders. The multipliers and adders scale each of the pixel photocurrents according to the convolution kernel being implemented. Each of

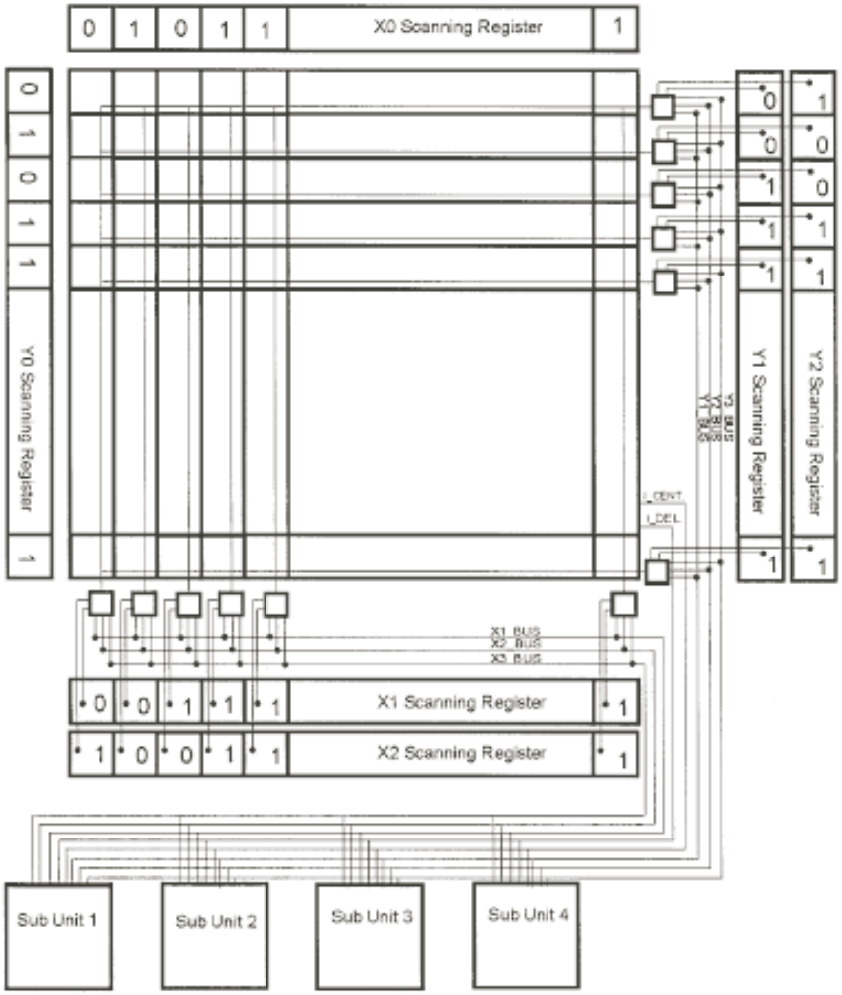


Figure 12: Overall architecture for Gruev's GIP: A prototype (16x16 pixel array) of the GIP has been fabricated in a standard 1.2μm CMOS process and its spatiotemporal capabilities have been successfully tested [50].

the four sub–processors can be independently programmed in parallel, allowing for different spatiotemporal convolutions to be performed on the incident image in parallel. A prototype (16x16 pixel array) of the GIP has been fabricated in a standard 1.2μm CMOS process and its spatiotemporal capabilities have been successfully tested. The pixel is composed of a photodiode, a nonlinear photocurrent amplification circuit, a sample-and-hold image delay element, and pixel selection switches. The pixel size was 30μm x 30μm, and the fill factor was 20%.

- Kawahito *et.al.* have developed a CMOS imager sensor with on–chip compression

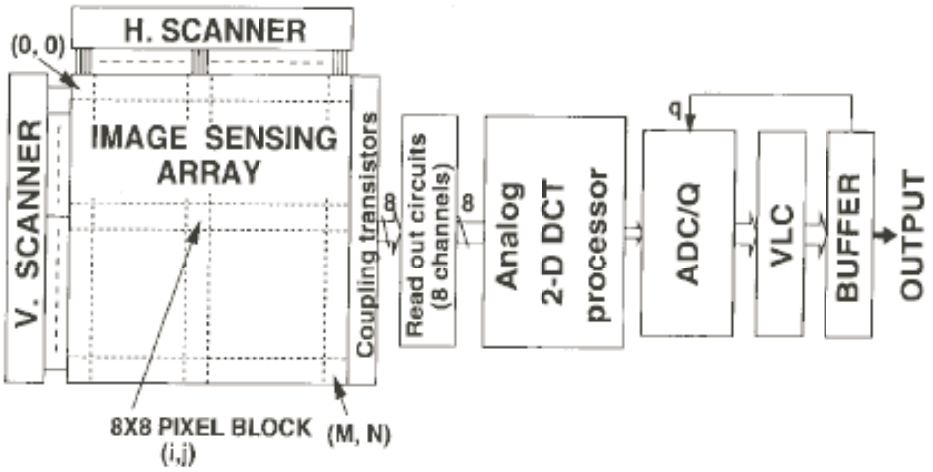


Figure 13: Chip schematic for DCT chip designed by Kawahito *et.al.*: The prototype was fabricated in triple–metal double–poly 0.35μm CMOS technology. The maximum peak signal–to–noise ratio (PSNR) was 36.7 dB. The pixel size was 16.1μm x 16.1μm, and the fill factor was 56.5% [67].

using an analog two–dimensional discrete cosine transform processor with a variable quantization level ADC [67]. Figure 13 shows the schematic of the proposed architecture. Fully differential switched–capacitor circuits have been used to design the 8x8 point analog 2-D DCT processor. The accumulated signal charge in the pixel is converted to a voltage using the readout circuits. The analog 2-D DCT processor computes in parallel as an unit of column. This 8x8 DCT processor consists of a 1-D DCT processor and an 8x8 analog memory cell. For computing the eight point DCT parallel data is read out eight times from the image sensor block. Two 2-D DCT cores are used for increasing throughput. The imager array has a dedicated eight channel parallel readout scheme. Quantization is performed by the variable quantization level ADC. Hence the compression ratio can be adaptively changed depending on the image quality and the type of image. This adaptive ADC is particularly useful for reducing the power dissipation of video ADCs. The authors claim that they could not get the whole system working and data from just the the 2-D DCT processor is presented in the paper. The prototype was fabricated in triple–metal double–poly 0.35μm CMOS technology. The maximum peak signal–to–noise ratio (PSNR) was 36.7 dB. The pixel size was 16.1μm x 16.1μm, and the fill factor was 56.5%. This unit achieves reasonable

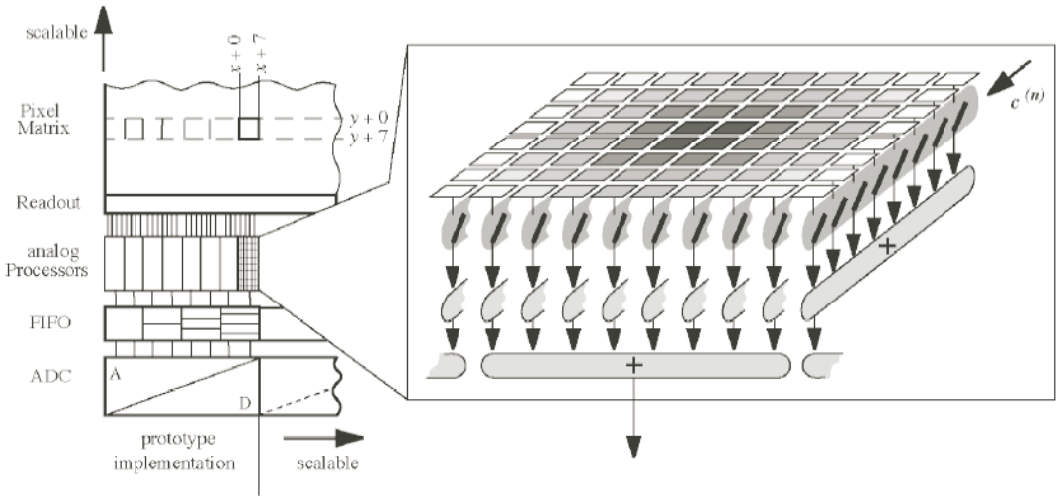


Figure 14: Chip schematic for design by Graupner *et.al.*: The chip was designed using 0.6μ CMOS technology. The pixel size was 12μm x 12μm with a fill factor of 48% [49].

quality but the flexibility is limited to the just performing 2-D DCT operations.

- Graupner *et.al.* have presented a single–chip integration of a CMOS sensor with embedded flexible processing array and dedicated ADC [49]. Figure 14 shows the schematic of the proposed architecture. The array can perform convolution and transform algorithms with arbitrary kernels. The kernels are stored digitally. The analog output from the photo sensors are multiplied with the stored kernels and the results are added. The processor unit operates on a potion of the image only. It is organized in slices of 8x8 elements corresponding to a square image block. Each slice consists of 64 analog memories, 64 bit–serial multipliers, and a multi–input adder. During operation, the image data of eight rows is transferred from the sensor array into the processing unit. For every applied bit level of the kernel, all the necessary multiplications and additions are carried out in parallel. Each processor computes one analog value per bit level. These are then fed to the ADC after being rearranged in a first–in first–out (FIFO) memory. The kernel coefficients are provided from off–chip components. After eight rows of the captured image have been copied into the processor, the kernels are applied subsequently. When all the kernels have been applied the computation is started again. The processing element consists of a current memory cell to

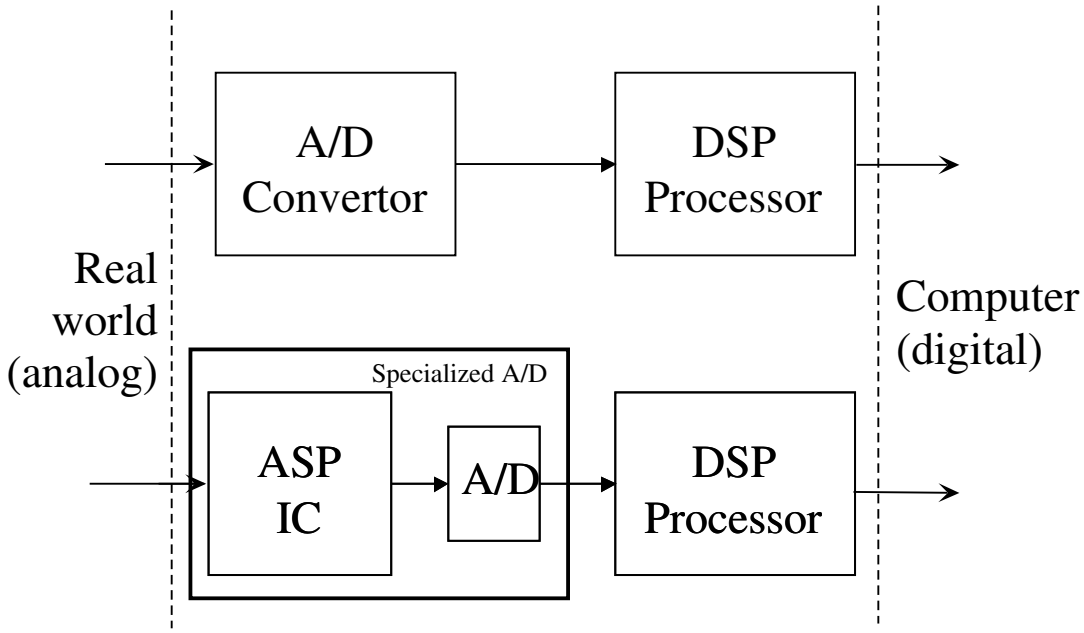


Figure 15: Illustration of the tradeoffs in cooperative analog/digital signal processing: We assume the typical model of signals coming from analog sensors that need to be utilized by digital computers. The inverse problem, digital signals going to real-world actuators, is similar in nature. One approach is to put an analog-to-digital (A/D) converter as close to the sensor signals as possible, and allow the remainder of the computations to be performed digitally. An alternate approach is to perform some of the computations using analog signal processing, thus requiring simpler A/D converters, and reducing the computational load of the resulting digital processors. One could group this analog computation and A/D converter as a specialized A/D converter that gives more refined information (Fourier coefficients, phonemes, etc.) than a literal map of the incoming signal. The question of where to put this boundary line strongly depends upon the particular requirements of an application.

store image data and two switches for multiplication. The prototype was fabricated in double–metal double–poly 0.6μ CMOS technology. The pixel is a standard three transistor integrating pixel cell comprising a diffusion well diode, source follower, a reset and a select switch. The pixel size was 12μm x 12μm with a fill factor of 48%. Although this architecture gives reasonably good data, the kernel has to supplied externally, the processor occupies large area while computing transforms just one block of the image at a time.

1.6 Cooperative analog/digital signal processing

Neither analog signal processing nor digital signal processing can exist in current technologies without the other; that is, real–world signals are analog while much of the modern control and communication is digital. Figure 15 shows two ways of addressing this

problem. Typically, one does not think of analog and programmability together—analog circuits are primarily used for front end processing, while programmability has been an exclusively feature of digital processors. However, new advances in analog VLSI circuits have made it possible to perform operations that more closely reflect those done in digital signal processing applications. Furthermore, analog circuits and systems can be *programmable*, reconfigurable, adaptive, and at a density comparable to digital memories [54, 55, 61, 77].

We define cooperative analog–digital signal processing (CADSP) as looking at the issues of using combinations of programmable analog signal processing and digital signal processing techniques for real–world processing architectures [4, 56]. Our goal in CADSP is to build systems that benefit from the advantages of both types of signal processing to make something better than the sum of its parts and to enhance the overall functionality of a system by utilizing analog/digital computations in mutually beneficial way. Therefore, one might wonder if we have both digital and analog signal processing available, how does one choose a particular solution for a given application. The question of where to partition the analog–digital boundary, as shown in Fig. 15, is still an open research question and depends on the application.

1.7 Motivation for MATIA

As energy-crucial portable and multimedia applications are becoming more popular the emphasis of VLSI design is shifting to high performance, low power, and low voltage techniques. For imager systems as pixel size increases the scan-read-out process is a fundamental limitation for high resolution/frame-rate imaging systems [32, 117]. Compression algorithms can be used to overcome the communication bottleneck [48]. Discrete cosine transform (DCT) and wavelet transform (WT), among various block transforms, are popular in image/video compression applications including the standards like JPEG, MPEG, H.261 and H.263 [69, 67]. Although these can be realized using digital methods, low power analog techniques enable these functions to be performed in a more space and power efficient manner.

For performing matrix transforms one needs to store the matrix coefficients on-chip and

perform multiplications while maintaining a high fill factor [50]. Although these can be done using digital storage and multiplication blocks, analog and floating gate technologies enable these functions to be performed in a much more space and power efficient manner. We present our fully programmable block MAtrix Transform Imager Architecture (MATIA) that can preform all of the above mentioned functions. The pixel has a fill–factor of 46%. The matrix coefficients, which are stored on–chip, can be programmed to any desired arbitrary values. It also consists of differential current mode vector–matrix multipliers (VMM), and row parallel readout circuitry for faster image capture. The chip consumes 80μW/frame of power for a supply of 3.3V.

Our new imaging architecture is made possible largely by advancements in analog floating–gate circuit technology and its applications [55, 77, 54]. Floating–gate devices in imaging can be used to eliminate fixed pattern noise and to enable programmable and adaptive signal processing applied toward the images [24, 13]. These circuits have the added advantage that they can be built in standard CMOS or double-poly CMOS processes.

As we had seen from the previous sections, there have been several implementations of transform and/or convolution single–system chips, but none of them have the flexibility of functionality that is provided by MATIA. It can perform block transforms of full/sub images at 25 frames per second (fps). This architecture can be used for various applications like stereo, motion estimation using optical flow methods, super resolution, implementation of spatio-temporal filters, DCT-based feature extraction, and front end encoders for low power transform based JPEG/MPEG systems. The resulting architecture is a data–flow structure that allows for continuous computation of these matrix transform operations. The architecture's scalability makes it feasible to compute large scale digital camera resolution images. Of all the implementations it is also the most power efficient architecture. Some of the novel features of this system are:

- It has a new pixel architecture which allows for focal–plane processing while maintaining a high fill–factor (46%).
- It has a true column parallel architecture. Hence all the sub–blocks along a row are

being processed at the same time.

- It is fully programmable because of the use of floating–gate circuits.
- It is low power consumption($V_{DD} = 3.3V$) of 80μW/frame
- The processing is entirely current–mode.
- It is capable of performing convolution and block transforms of images using the same architecture.
- It can easily be scaled to other processes.

The thesis is organized into six chapters. In Chapter 2, we present an overview of floating–gate devices, and present an adaptive algorithm for programming floating gates accurately over a wide range of currents. In Chapter 3, we present the basic pixel elements and their characterization as well as the mathematics needed to predict the performance for a given application based on experimental measurements, including estimates of noise, speed, etc. In Chapter 4, we present the design of a four quadrant current–mode vector matrix multiplier. We present the concept and also characterization data from a test chip. In Chapter 5, we present the imager system architecture. The various sub–blocks and their interconnections are described here. We show system data of various on–chip image transforms. We also show the implementation of a low–power JPEG compressor using MATIA. We compare this implementation with a FPGA–only implementation for power comparisons. In Chapter 6, we describe some future directions in which these MATIA chips can be used and also conclude with the impact of this work.

CHAPTER II

ADAPTIVE PROGRAMMING OF FLOATING GATES ARRAYS

2.1 Introduction

Floating gates have been used as computational and memory elements in various applications that include adaptive circuits [18, 92], mass storage [99], data converters [65, 113], imagers [7, 25], etc. They have been used to store digital information for extended periods in structures such as EPROMs, EEPROMs, and flash memories. Floating gates can retain analog data for long periods, and the reported retention accuracy is about 6 bits for 15 years at more than 125°C [1, 92].

Programming of floating gates can be done using Fowler-Nordheim (FN) tunneling or hot–electron injection processes [25, 79, 92, 99, 113]. Pulse width modulation methods have been used to program floating gates [70] but accuracy is limited by the minimum pulse width that can be used for programming [72, 71]. In these methods a trade–off between accuracy and pulse width is observed. The total program time, using FN tunneling, increases with higher precision because of the logarithmic behavior of the mechanism [71]. Hot-electron injection can be used to program these memory elements with higher accuracy at very fast rates without the use of any special process.

We present a novel method that can be used to precisely program large floating–gate arrays with more than 99.8% of accuracy over a wide range of currents (more than 3.5 decades). This algorithm modulates the drain–to–source voltage of the pFET floating–gate transistor for a constant pulse width. Before each update, the algorithm finds the most optimal V_{DS} to be used for accurate programming of that element. In this paper we present measured data from arrays of floating gates that were fabricated in 0.25μm and 0.5μm N-well CMOS processes.

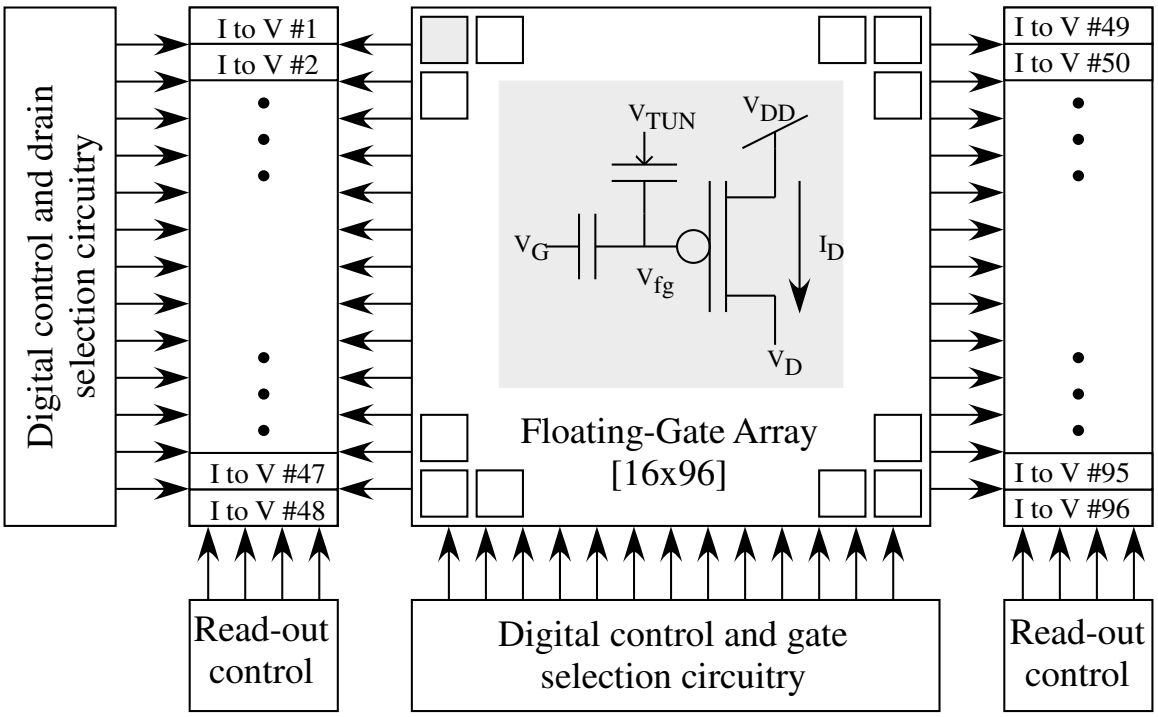


Figure 16: **Chip schematic for programming of floating–gate arrays**: The chip consists of a 96x16 array of floating–gate elements, peripheral digital control for isolation of floating–gate elements during programming, and row parallel readout circuitry.

This chapter consists of seven sections. Section 2.2 describes the chip architecture that was used for testing the proposed algorithm. Section 2.3 describes floating gates and programming methodology. Section 2.4 illustrates the proposed algorithm. Test setup and experimental data are discussed in sections 2.5 and 2.6, respectively. A simplified model of the proposed algorithm is presented in section 2.7. Finally the relevant aspects of this paper are summarized in the conclusion.

2.2 Chip architecture

The block diagram of the chip is shown in Fig. 16. The chip consists of a 96x16 array of floating–gate elements, peripheral circuitry for accurate programming, and an array of current readout circuits. The floating–gate transistors have the dimensions (W/L) of $6\lambda/4\lambda$. All the tunnel junctions were connected together so that FN tunneling can be used as a global erase.

Decoders and switching circuits are used for digital control and isolation of individual

floating gates for accurate programming. In program mode, we can easily reconfigure this circuitry on the outside edges for programming, resulting in very high circuit density. In the operation mode the floating gates can be used collectively for different applications. In this mode the peripheral programming circuitry is turned off and the floating gates are connected to their respective operation mode circuits. Integrating type I–V are used for reading the currents.

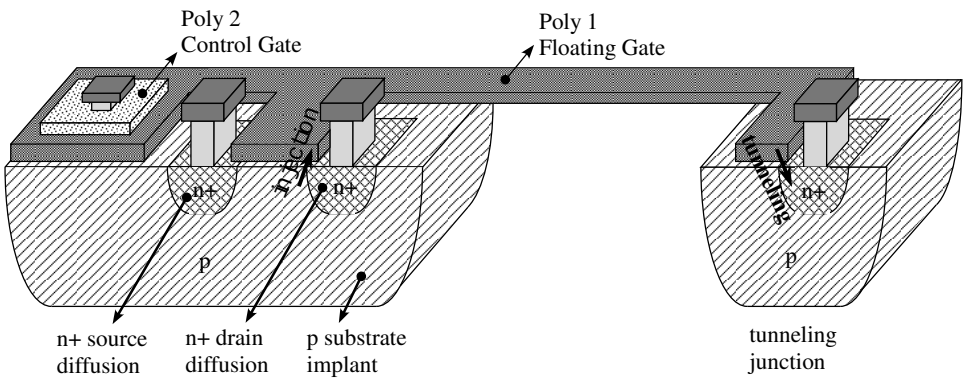
2.3 Floating gates

The cross sectional diagram of a floating–gate (FG) pFET is shown in Fig. 17(a). A floating gate is a polysilicon gate surrounded by silicon dioxide [76], a high quality insulator. This allows charge on the floating gate to be stored permanently, providing long term memory. The floating–gate voltage is determined by the charge stored on the floating node. It modulates a channel between a source and drain, therefore can be used in computation. The current of this transistor will be determined by the capacitively coupled input voltage and the floating–gate voltage. This floating gate can be the gate of a MOSFET and can be capacitively connected to other layers. A floating–gate transistor is programmed by adding or removing charge from its floating node. The charge stored on a FG transistor can be modified using the following methods:

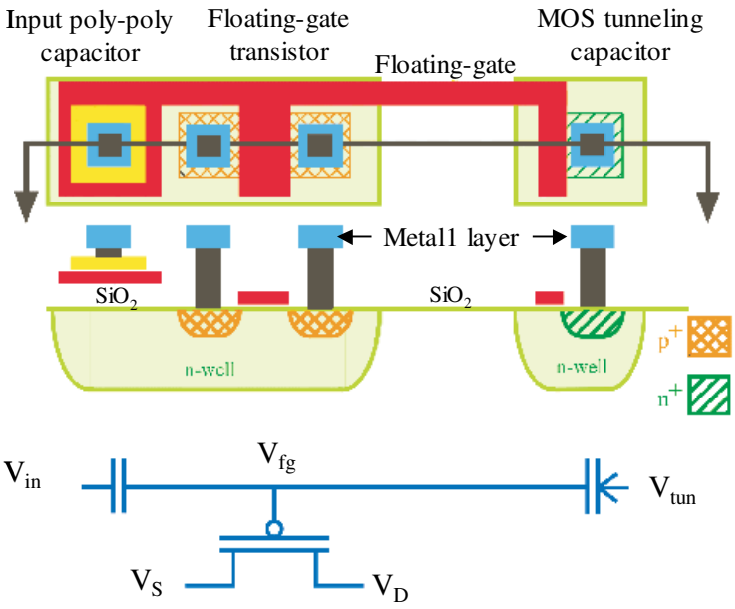
- **UV photo injection**: The classical method for modifying the charge on a FG is using short–wave ultra violet light, known as UV–C. Exposing silicon dioxide to UV–C will impart enough energy to some carriers to surmount the silicon–dioxide energy barrier. This method has been used extensively in the case of memory elements. It is perfect for normalizing arrays. Some of the drawbacks of this method are:

  1. Requires costly package
  2. Programming time is lengthy
  3. Difficult to isolate and selectively program individual elements

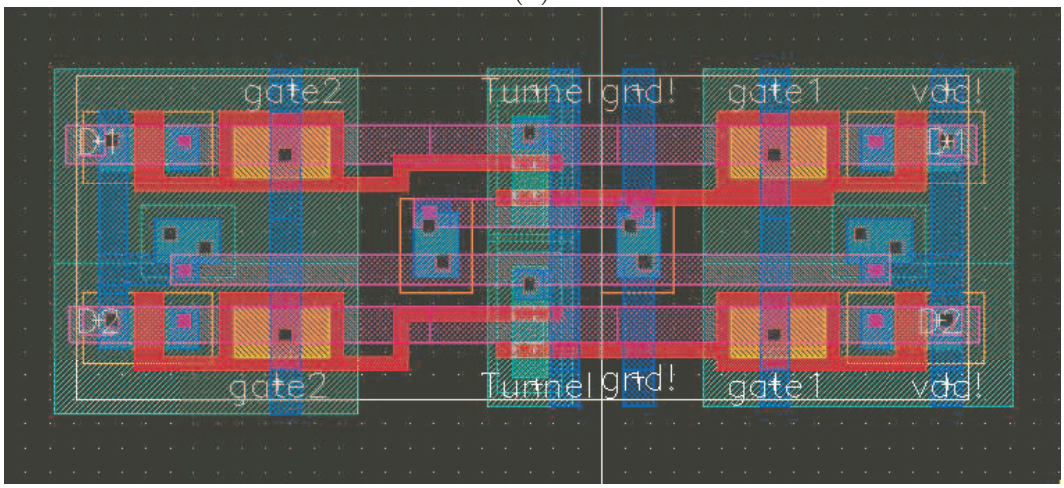
- **Electron tunneling**: FN tunneling [40, 79] is used to remove electrons from the floating gate. The tunneling junction is schematically represented by a capacitor,



(a)



(b)



(c)

Figure 17: Floating–gate element: (a)Cross section of a floating–gate (FG) transistor. The FG transistor is a standard pFET transistor in a N-well process, with its gate connected to a poly-poly capacitor and a MOS capacitor. This allows charge to be stored at this node, providing long term memory. The floating–gate charge can be changed by using tunneling and hot–electron injection processes, (b) Shows the layout, cross–sectional view, and symbol of single FG, (C) Shows the layout of a 4x4 matrix of FG transistors.

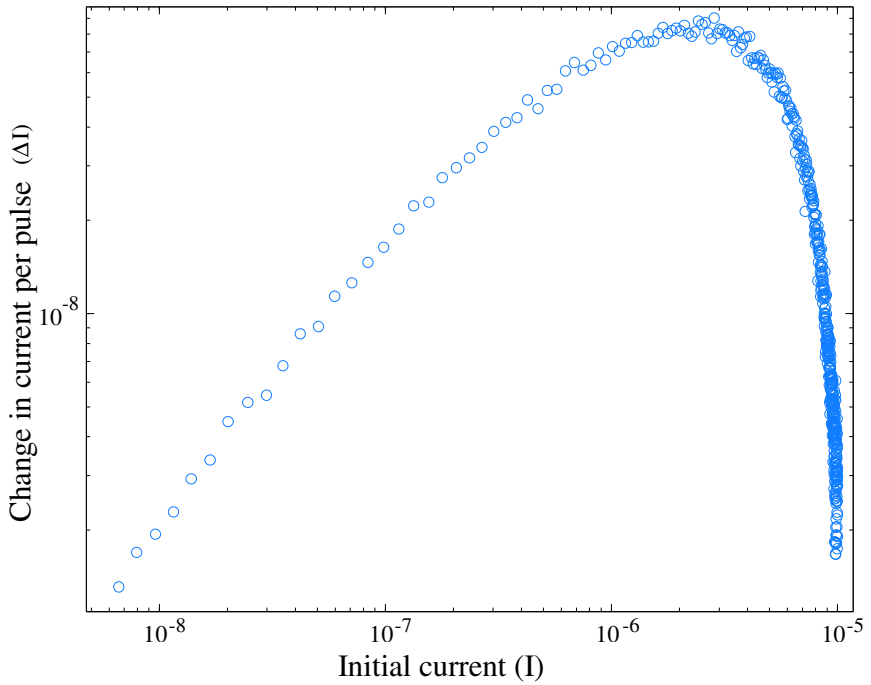
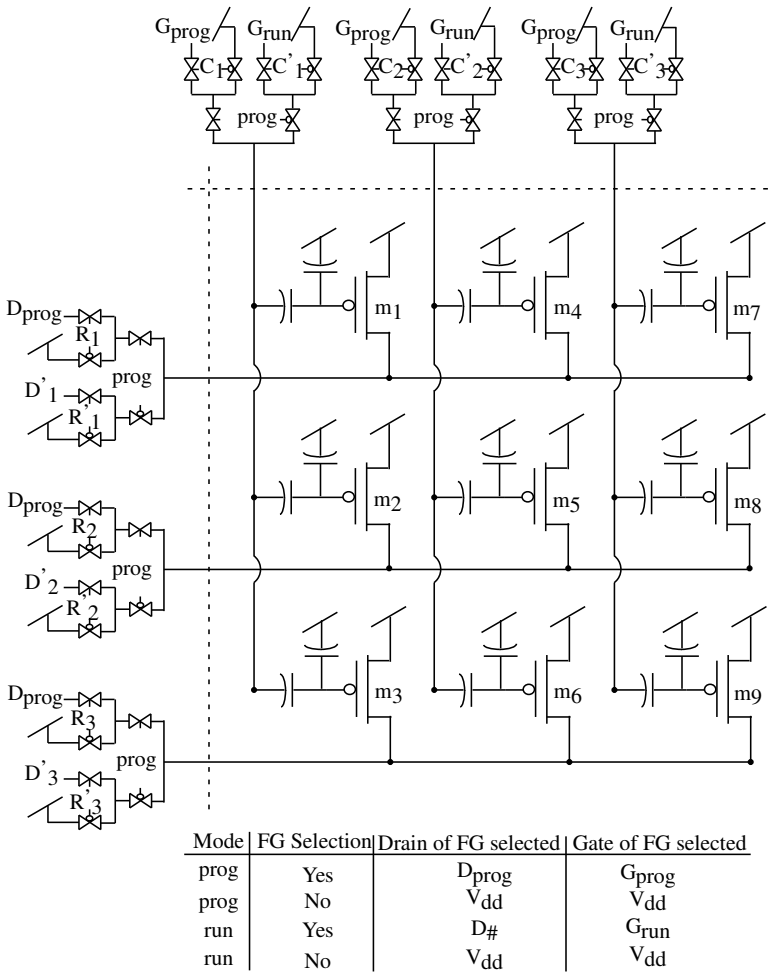


Figure 18: Injection rate: A plot of change in drain current per pulse vs initial current. A 100μs pulse width and a 3.8V V_{DS} where used. The rate falls drastically after 2μA for a transistor with a W/L of 1.8λ/1.2λ in a 0.25μm process.

which couples the tunneling voltage terminal to the floating gate, as shown in Fig. 17(a,b). The arrow on the capacitor denotes the charge flow. Increasing the voltage across this tunneling capacitor, either by increasing the tunneling voltage or decreasing the effective floating–gate voltage, increases the effective electric field across the oxide, thereby increasing the probability of the electron tunneling through the barrier. The strength of the field required for tunneling to occur depends on the thickness of silicon–dioxide. An inherent problem of FN tunneling is the texture of most insulators growing on silicon resulting in variable thickness. Since FN tunneling is an exponential of both the field and the silicon–dioxide thickness, most of the current flows through the thinnest area. These local spots are called "hot–spots". The currents can be so high that the silicon's diamond shaped lattice breaks down, leaving open traps for free carriers. The long term effect of these is called "wear–out" which makes silicon–dioxide leaky. Tunneling selectivity along a row in an array is entirely a function of how far apart the two floating gates are pushed by the gate inputs. This is due to the



Mode	FG Selection	Drain of FG selected	Gate of FG selected
prog	Yes	D_{prog}	G_{prog}
prog	No	V_{dd}	V_{dd}
run	Yes	D#	G_{run}
run	No	V_{dd}	V_{dd}

Figure 19: Isolation of floating gates: This schematic shows how floating–gate elements can be isolated from an array for accurate programming. Peripheral digital control has been used for isolation hence high density arrays are feasible.

fact that the amount of tunneled current is based on the voltage across the tunneling capacitor. Typical tunneling voltages (V_{tun}) for a 0.5μm and a 0.25μm process are 16V and 7V, respectively.

- **Hot–Electron Injection**: Hot–electron injection [23, 107] is used to add electrons to the floating gate. In order for injection to occur two conditions must be meet, a high current flowing through the transistor and a high gate to drain electric field. The injection rate will be determine by the V_{ds} voltage and the pulse width used. The hot-hole impact ionization (in a pFET carrier are holes) creates electrons at the drain edge of the drain to channel depletion region due to the high electric fields there. The

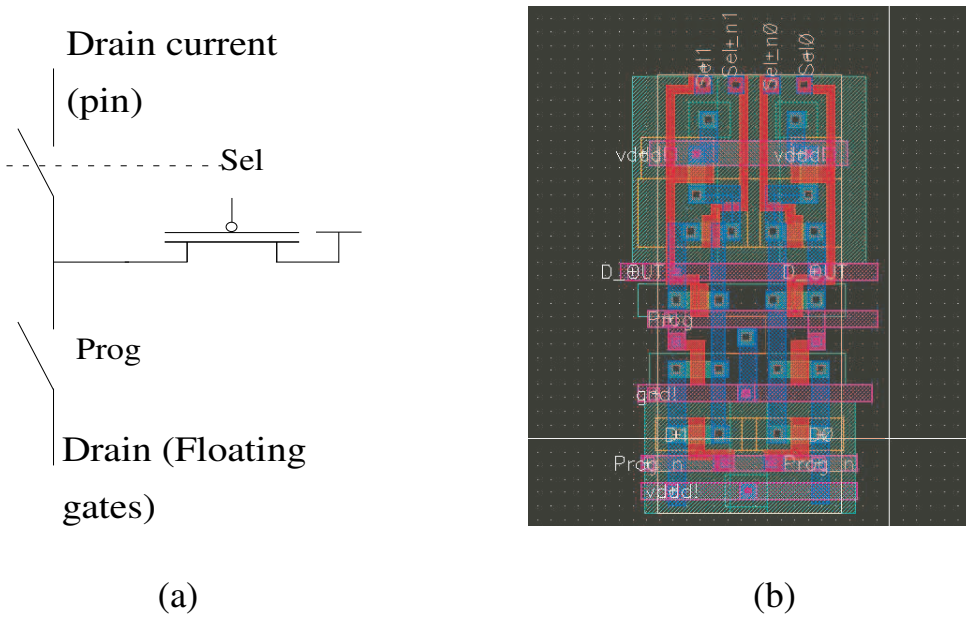


Figure 20: Drain/gate isolation for programming: (a) Schematic, (b) layout (2 cells) of isolation multiplexors used for isolation of floating gates for accurate programming.

hole impact–ionization current is proportional to the pFET source current (I_s) and is the exponential of a smooth function of the drain-to-channel potential (ϕ_{dc}). These electrons travel back into the channel region, gaining energy as they go. When their kinetic energy exceeds that of the silicon–dioxide barrier, they can be injected into the oxide and transported to the floating gate.

We increase or decrease the output current of the floating–gate device by using hot-electron injection [23] or electron tunneling [79], respectively. To perform injection of a floating–gate transistor, V_{DD} (1.8V and 3.3V for the 0.25μm and 0.5μm processes, respectively) is increased to a HIGH V_{DD} (3.8V and 6.5V for the 0.25μm and 0.5μm processes, respectively). All other voltages are also increased with respect of V_{DD}. This process will be referred to as ramping up; its counterpart will be referred as ramping down. To get the high currents necessary for injection to occur, the drain voltage (V_D) is then pulsed to a lower voltage for a certain amount of time (t_{pulse}), thus creating a high V_{DS} voltage. Typical V_{DS} voltages used for injection ranges from 4.0V to 6.5V for a 0.5μm process. After injection is completed all voltages are restored to their original values (ramped down). The electron injection rate is a function of t_{pulse} and the V_{DS} voltage when pulsed.

The injection rate of a pFET, in 0.25μm process, is shown in Fig. 18. Here a pFET is successively pulsed, using same the source–drain voltage for the same duration for each pulse. The change in current is plotted versus initial current on a log–log scale. It can be observed that the rate increases for subthreshold currents and decreases for above threshold currents. For this transistor the rate peaks at about 2.8μA.

In our scheme, devices are precisely programmed using hot-electron injection while a global erase is achieved with tunneling. Figure 19 illustrates how each floating gate can be isolated from an array for precise programming. The isolation circuitry is made of multiplexors that switch the drain and gate voltages of the desired element onto a common bus for each signal. Other elements are switched to a separate voltage to ensure that those devices do not inject. The table shows the different operations and how each gate and drain lines are switched for isolation and accurate programming. This scheme allows for simultaneous reading or computation along a column. Figure 20 shows the multiplexors that were used for isolation. They were designed such that they can be tiled seamlessly and also have a pitch of 45λ (13.5μm) in one direction.

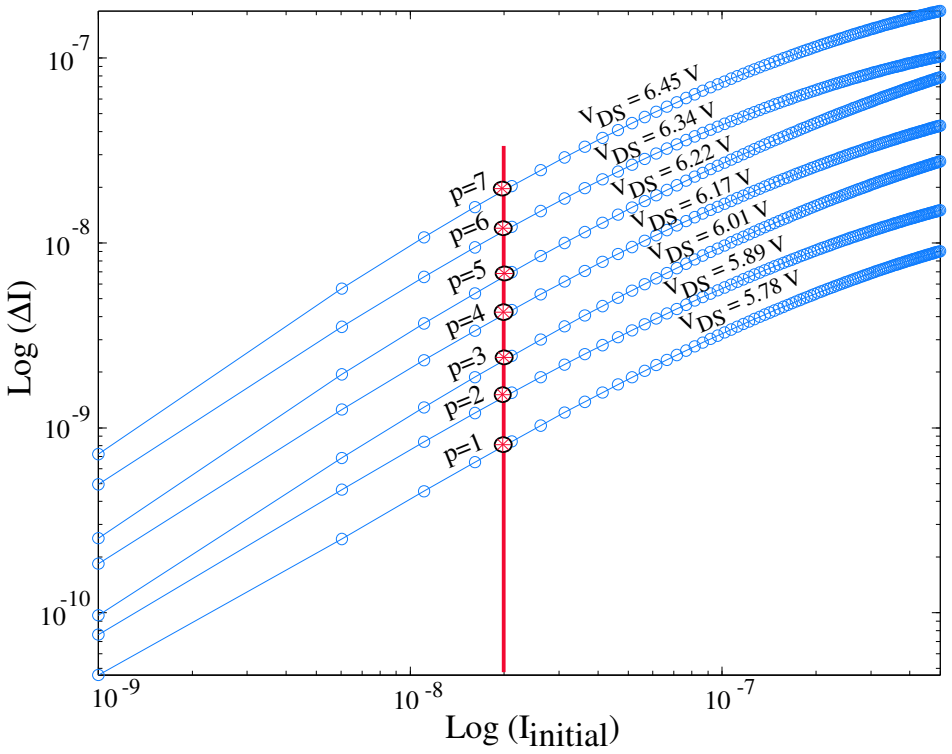
2.4 Novel programming scheme

The predictive algorithm uses both V_{DS} control and pulse width modulation to perform accurate and fast programming of large arrays of floating gates. We perform two types of calibration on the chip. The first is the V_{DS} calibration for fixed pulse width (20μs), while the second one is the pulse width calibration for fixed V_{DS} (6.5V). These enable us to predict the required V_{DS} and pulse width for accurate and fast programming over a wide range of currents (both sub–threshold and above threshold).

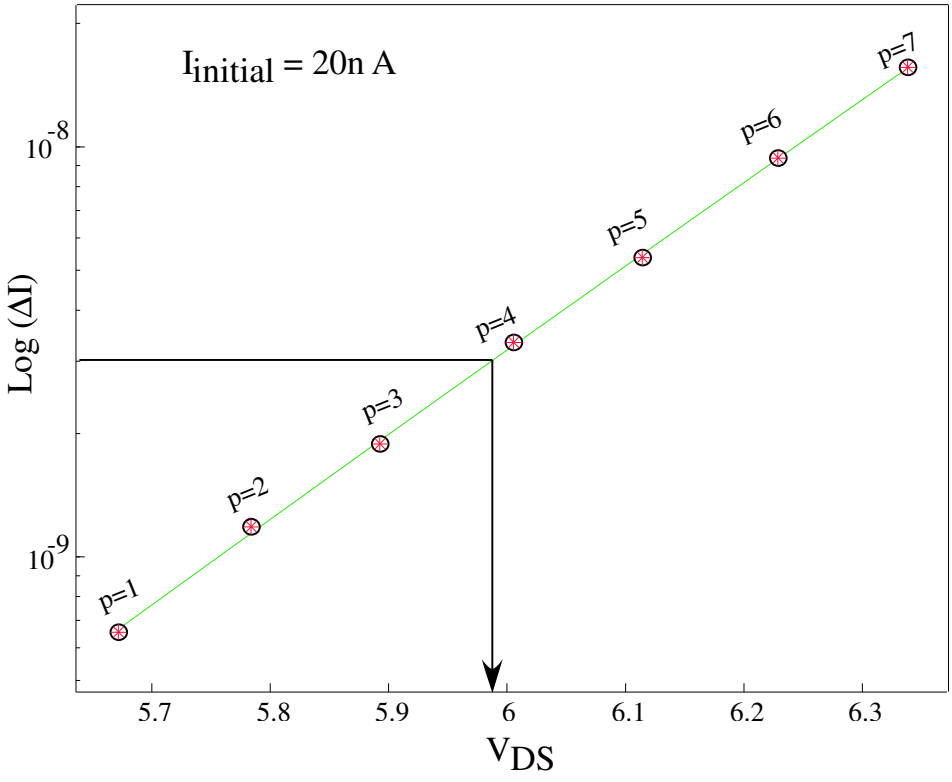
Pulse width modulation is used for bringing an element closer to the target initially if the element is very far off from the target, after which V_{DS} control is used for fine tuning of injection rates. The pulse width is increased only if the calculated V_{DS} is more than the HIGH V_{DD}.

2.4.1 Calibration procedure

For the V_{DS} calibration the following is performed:



(a)



(b)

Figure 21: Characterization curves: These plots show the characterization that was conducted for V_{DS} calibration. (a)Shows the variation of injected currents for different initial currents as a function of different V_{DS}. (b)A plot of change in current for different V_{DS} for $I_{initial}$ = 20nA. This plot is obtained from plot (a).

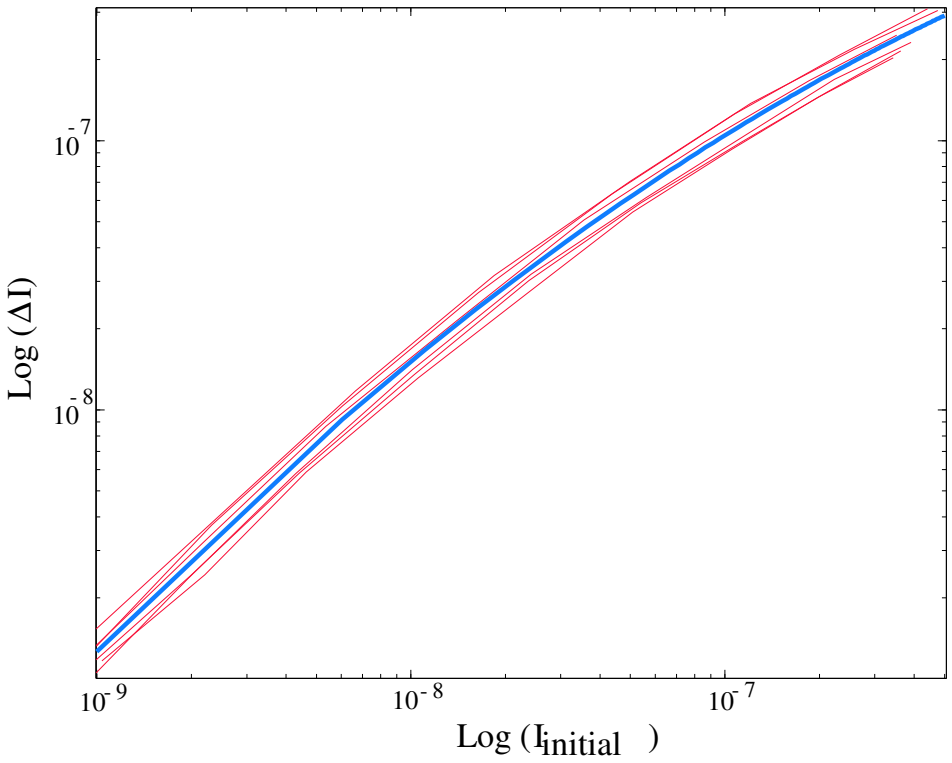


Figure 22: Measured injection rates: The measured injection rates are shown for 6 elements (thin lines) for a given V_{DS}. The average regressed rate for this V_{DS} is given by the thick line. The regression was obtained from ten elements (six of which are shown here for clarity) for the same pulse width and same V_{DS}.

1. The chip is ramped up.
2. Each element is pulsed (injected) once for the given V_{DS} and a constant t_{pulse}.
3. The chip is ramped down.
4. The current of each element is read and stored.
5. Steps 1 to 4 are performed for the same V_{DS} until the measured current exceeds an upper threshold (500nA).
6. Steps 1 to 5 are then performed for various V_{DS}.
7. The change in current vs initial currents ($I_{initial}$) is plotted for different V_{DS}. A second order fit is used to regress the data. Figure 21(a) shows the output for one of the calibrations. A first order fit would suffice for sub–threshold currents but for above

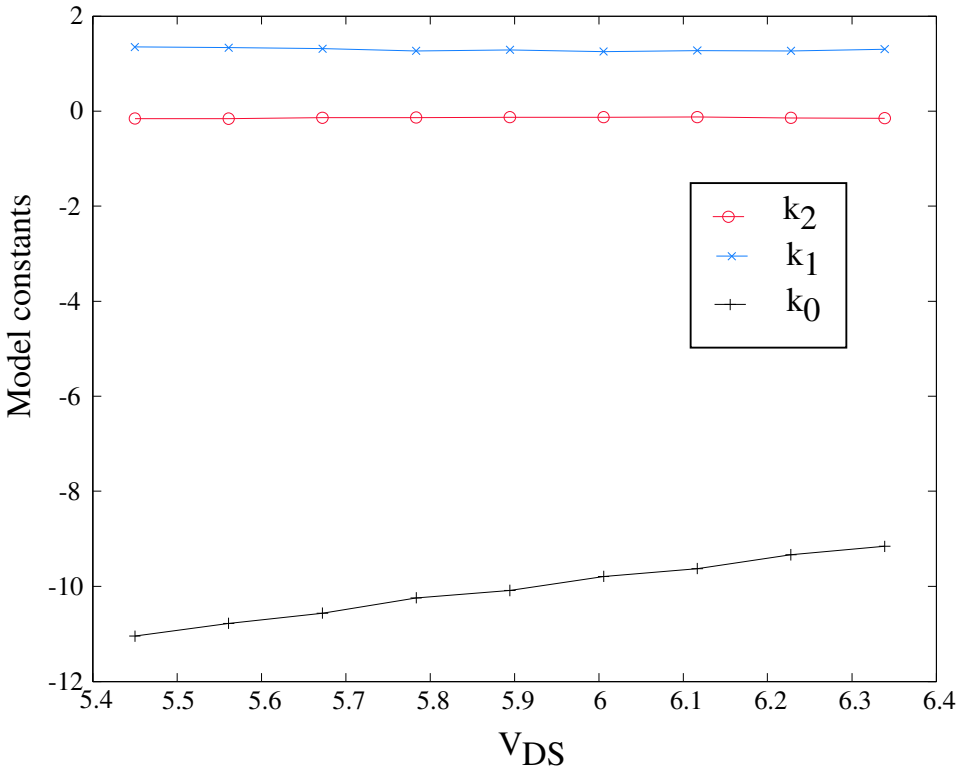


Figure 23: Variation of model constants: The variations of k_0, k_1, and k_2 with V_{DS} are plotted.

threshold currents a second order gives a better estimate, as the injection efficiency goes down for above threshold currents as shown previously in Fig. 18.

Pulse width calibration is performed the same way as V_{DS} calibration but with changing pulse width and fixed V_{DS}. Calibrations are conducted for elements spaced across the array so that process and size mismatches average out. This takes care of run to run and chip to chip mismatch of different injection rates for different floating–gate elements.

The calibration was conducted for 10 elements for each regressed curve as shown in Fig. 22, but it can be conducted for lesser number of elements. Here measured data from six elements are shown for clarity. The data was regressed using a second order fit (thick solid line). Since observed mismatches in the injection rate curves are small, an asymptotic approach precludes the need for calibration of every element. Figure 24 shows the variation of log(ΔI) vs V_{DS} for various initial currents. It is observed that for a given V_{DS} injection rate increases with increasing initial current. The family of curves are parallel to each other

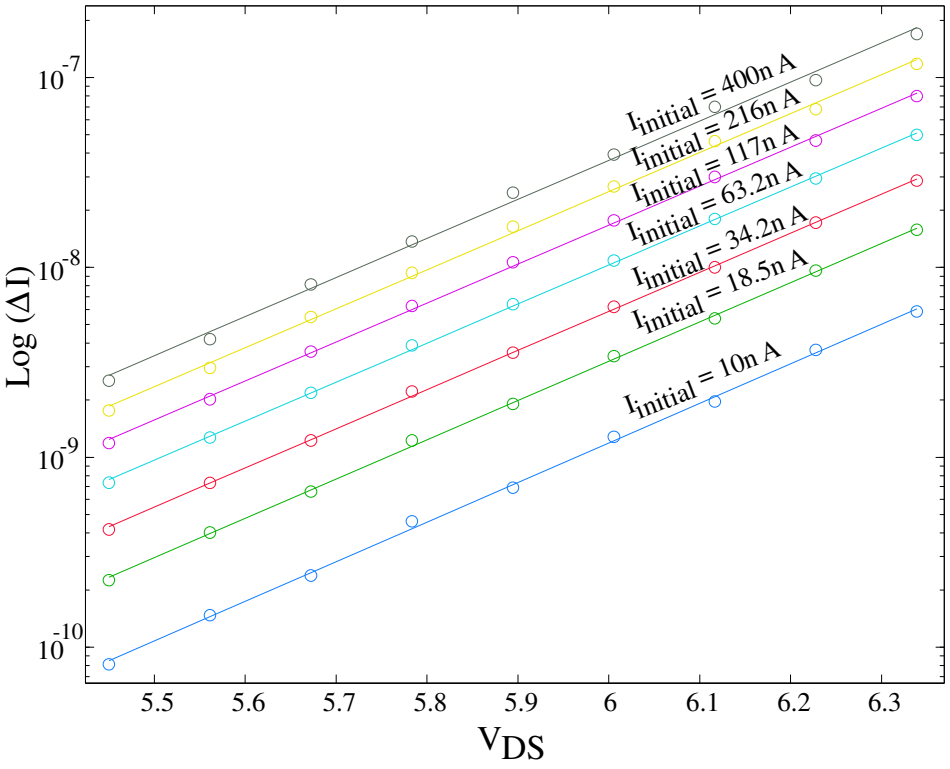


Figure 24: Variation of change in current log(**ΔI**) **with** V_{DS} **for various** $I_{initial}$: The bubbles indicate measured data, while the lines show a first order approximation for each $I_{initial}$. It can be observed that the slopes of the curves are constant, while the intercept depends on $I_{initial}$.

and the intercept depends on the initial current. A linear regressed fit can be used to model this variation, as observed from the solid lines on Fig. 24.

Mathematically, the variation of log(ΔI) with $I_{initial}$ for various V_{DS} has been modelled as follows:

$$\log\left(\frac{\Delta I}{I_X}\right) = k_2\left[\log\left(\frac{I_{initial}}{I_X}\right)\right]^2 + k_1 \log\left(\frac{I_{initial}}{I_X}\right) + k_0 \tag{1}$$

where k_0, k_1, and k_2 are unitless functions of V_{DS} (as can be observed on Fig. 21(a)), and I_X is a scaling factor with units of Amperes. The variations of k_0, k_1, and k_2 with V_{DS} are plotted in Fig. 23. It can be observed that k_0 is a linear function while k_1, and k_2 are almost constants.

From Fig. 21(a) the variation of log(ΔI) with V_{DS} for various $I_{initial}$ has been plotted in Fig. 21(b). This can be modelled as a linear function:

$$\log\left(\frac{\Delta I}{I_X}\right) = mV_{DS} + f \tag{2}$$

where m is the slope (unit of 1/Volts), f is the intercept (unitless), and I_0 is a scaling factor with units of Amperes, for a given $I_{initial}$. Both m and f are functions of $I_{initial}$ (as can be observed from Fig. 24.

2.4.2 Adaptive programming

The adaptive programming procedure is as follows:

1. Measure initial currents for the array to be programmed.

2. For each current and the target current the optimal V_{DS} is calculated using the calibration data already obtained. From Fig. 21(a) for each $I_{initial}$ the variation of log(ΔI) vs V_{DS} is obtained. This data is then regressed using (2). An optimal V_{DS} is then calculated for the required ΔI as shown in Figure 21(b). For example, for an initial current of 20nA, a V_{DS} of 5.98V is required for a 2nA change (Fig. 21).

3. If the calculated V_{DS} is more than HIGH V_{DD} then pulse width modulation is used to obtain higher injection rates. The optimal pulse width required can be obtained from the pulse width characterization curves.

4. The chip is then ramped up to a voltage of HIGH V_{DD} so that the drain can be pulsed to the calculated V_{DS}.

5. Each element in the array is then injected individually by changing the drain-to-source voltage to the calculated V_{DS}.

6. The chip is ramped down to the working V_{DD}.

7. The currents are measured. If measured currents are less than the respective target currents then steps 2 to 7 are performed until the element has been injected to the desired current.

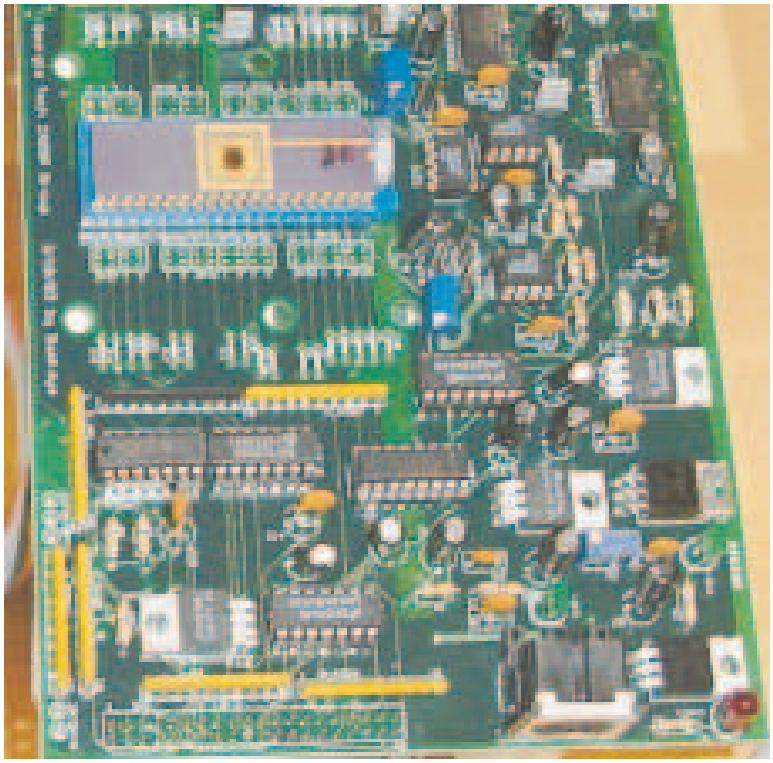


Figure 25: Test setup for programming: This printed circuit board was fabricated to facilitate experimental measurements. The board, which is digitally controlled with an FPGA, provides the necessary digital and analog voltages for the operation of the chip.

Since there would be an error associated with injection at higher V_{DS}, the injection process is performed in stages so that the target current is reached in a maximum of ten steps. This ensures that larger V_{DS} is used for moving the chosen element closer to the target when it is very far off (when required percentage change $> 100\%$) from the target while smaller V_{DS} is used when the element is very close to the target current, for better accuracy. For each step a target current is set depending on the difference of $I_{initial}$ and I_{target}. The algorithm predicts the required V_{DS} for each element for each stage of injection.

2.5 Test setup

A four layered printed circuit board (PCB), shown in Fig. 25, was designed to test this chip and validate our algorithm. The PCB provides the necessary analog voltages and level shifted digital signals for programming. It also supplies the analog signals for the chip operation. The analog voltages are generated using 12-bit DACs, while the I-V voltage

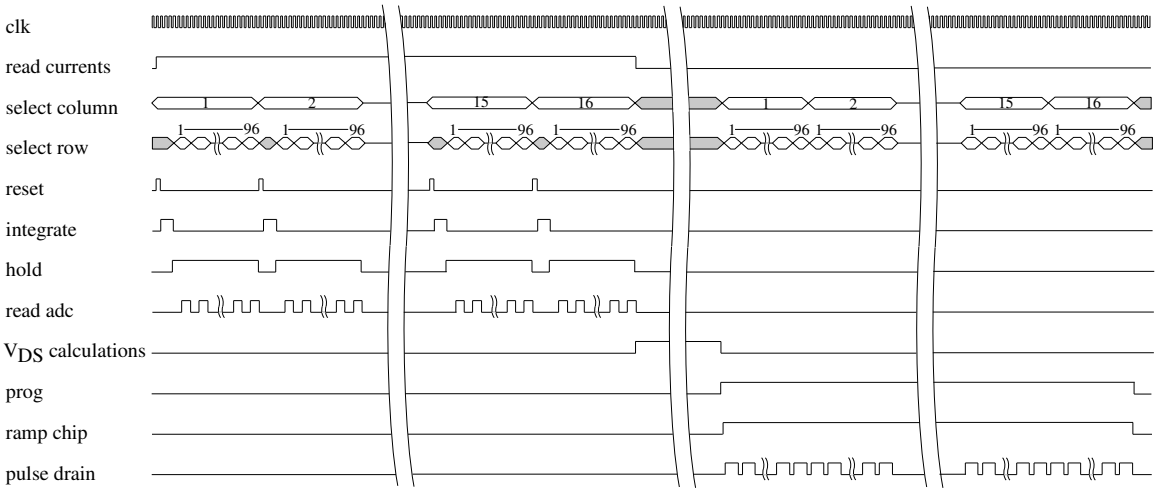


Figure 26: Timing diagram: Sequence of steps followed during one programming cycle. First, all the currents on the array are monitored. Then the necessary V_{DS} voltage is calculated for each element in the array. Finally, the whole array is pulsed with its respective calculated V_{DS}. This process is repeated until each element is programmed to the desired value.

outputs are obtained through 10Msps 14-bit ADCs. All the digital control signals coming from the field programmable gate array (FPGA) are level shifted so that the same board can be used for testing chips from different technologies.

The PCB is controlled using a commercially available FPGA. Communication with the personal computer (PC) is available through a 100Mbit Ethernet connection. A MATLAM^{TM} interface that provides a direct link to the FPGA has been developed. The FPGA provides digital control of all devices on the PCB and also generates high speed and accurate controlled signals needed for programming.

Figure 26 shows the timing diagram of the chip for one cycle (pulse all floating gates once) of programming. The floating gates are first chosen and the respective currents are then measured in a parallel fashion. The currents for all the floating gates along a single column are integrated at the same time on their respective I-Vs. During the hold time all the voltages are read out in a sequence. After this the next column is selected, the I-Vs are reset, and the procedure is repeated for current readout. After all the currents have been readout the respective V_{DS} are calculated from the characterization data. The chip is then configured for programming and is ramped up to HIGH V_{DD}. The floating gates are then chosen one by one, and are pulsed sequentially for programming each to their desired values.

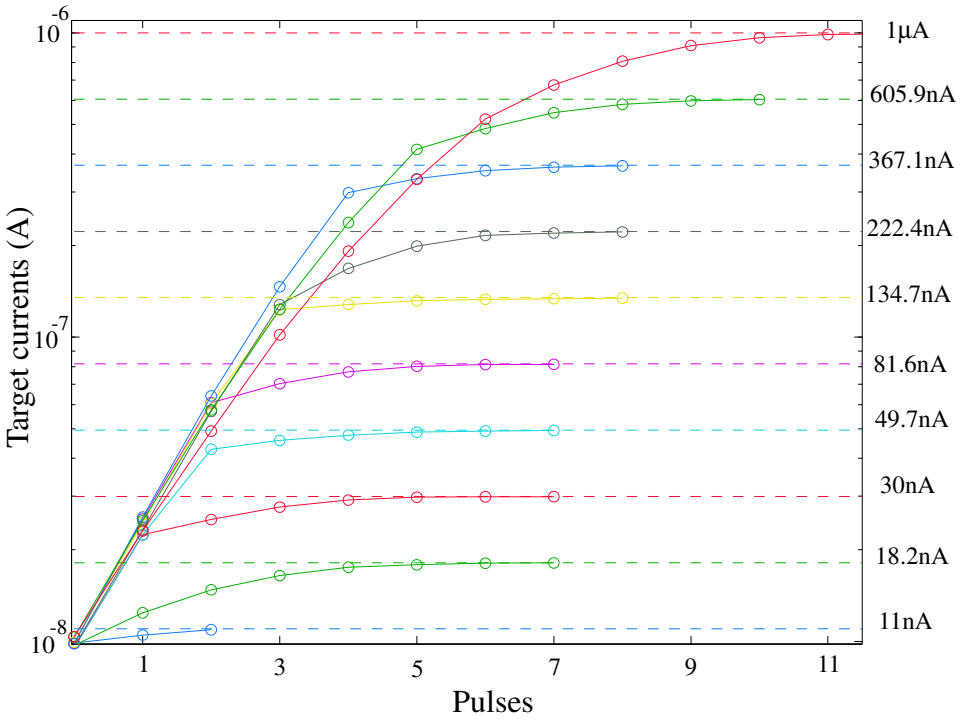


Figure 27: Accuracy of adaptive programming : Plots showing asymptotic approach towards different target currents. The dash lines are the target currents, while the bubbles indicate injected currents after each pulse. The average number of pulses required to hit a target current within two decades is 7-8.

After the whole array has been pulsed once, the chip is ramped down, and the currents are readout again for comparison with the target values. If a floating gate has already reached the target value then it is not pulsed during the next cycle.

2.6 Measured results

The algorithm was tested using large floating–gate arrays in 0.25μm and 0.5μm N–well CMOS processes. The chips were calibrated for different drain to source voltages. Different elements were chosen for each calibration.

Figure 27 shows how ten different elements were programmed asymptotically using the proposed algorithm. Each element had an initial current of 10nA (an arbitrary choice). The dashed lines show the target currents. The programming procedure stops when an element has been programmed to within 0.2% of the target current. The number of pulses increase with increasing target currents. The average number of pulses required are 7-8 for programming currents within 2 decades.

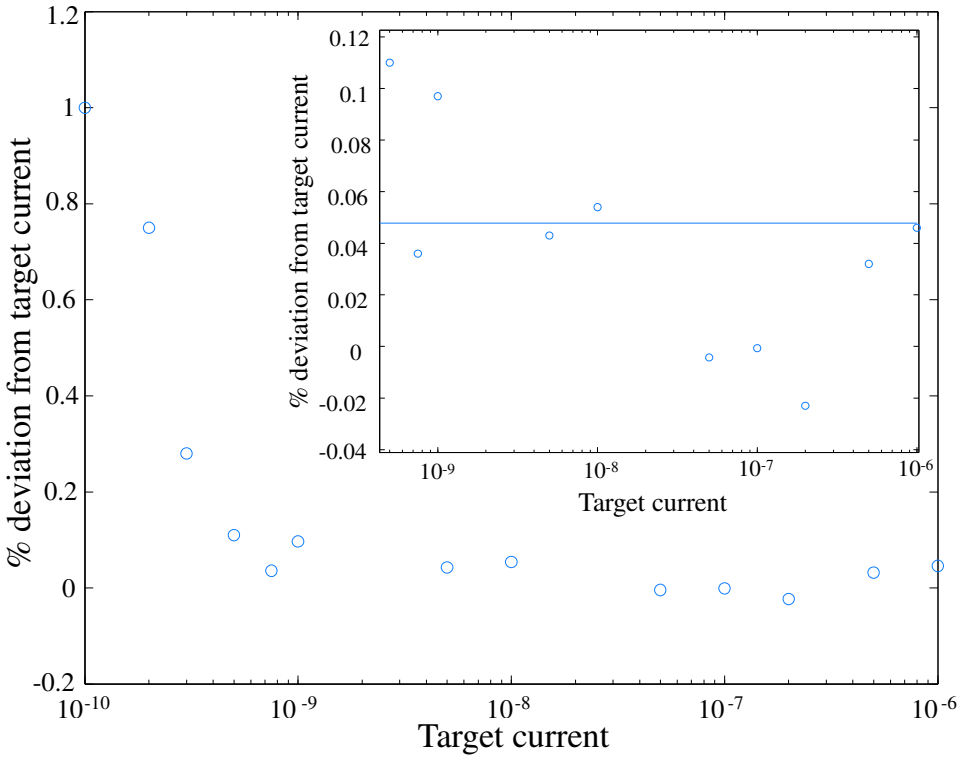
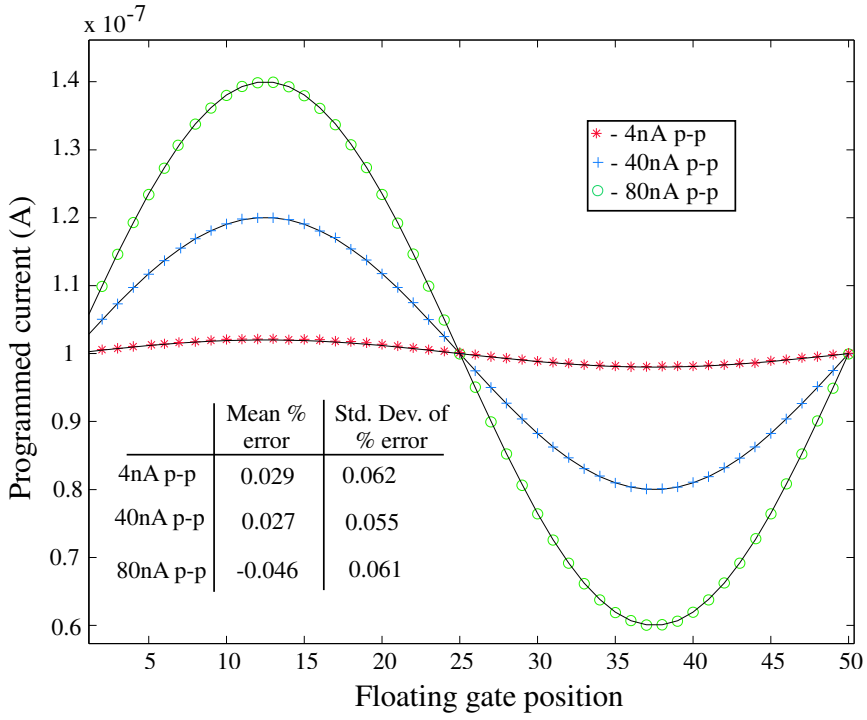


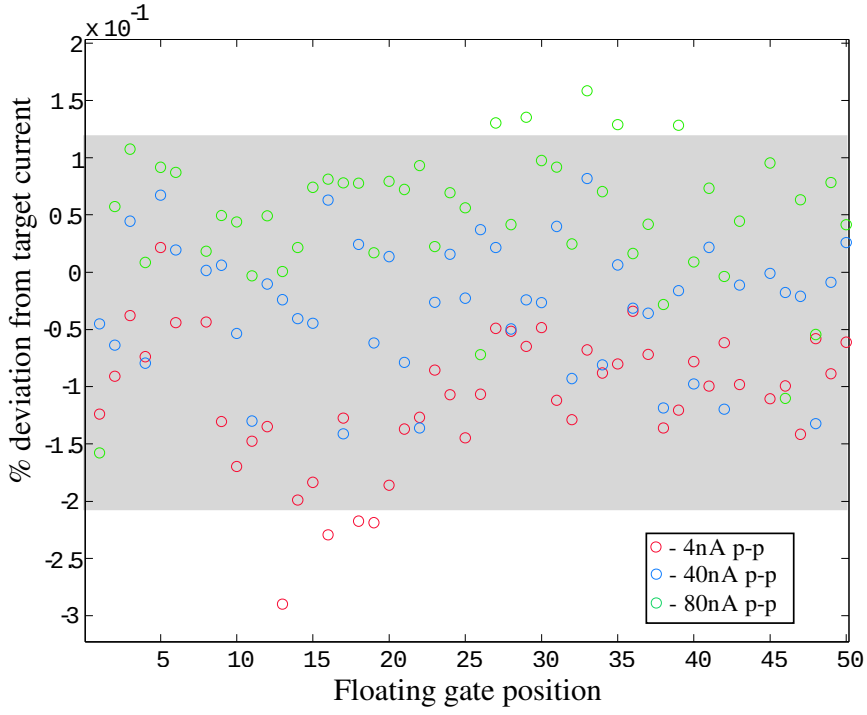
Figure 28: Percentage error for adaptive programming: The plot shows the percentage error for a wide range of currents. The inset shows the same for 3.5 decades of programmed currents. The average error was 0.05% while the maximum error was 0.12%

This algorithm can be used to program both deep sub–threshold currents and above threshold currents. The average percentage error (for 50 samples for each target current) for the whole range is plotted in Fig. 28. The algorithm can be used to program with more than 0.2% of accuracy over 3.5 decades of target currents (as shown in inset) in both 0.25μm and 0.5μm N-well CMOS processes. This accuracy is limited by the resolution of the on–chip I–Vs used for measuring the currents and also by the accuracy of the DACs used for pulsing the drain.

Floating–gate architectures have been used as analog computational elements in a number of applications. Arbitrary waveforms can be stored on–chip accurately using the proposed algorithm. Figure 29 shows sine waves of 4nA p–p, 40nA p–p, and 80nA p–p that were programmed onto 50 floating gates. The DC of these sine waves were 100nA. The bubbles indicate the programmed values while the solid lines represent the ideal targeted sine waves. The pulse width used for this experiment was 20μs. The average number of

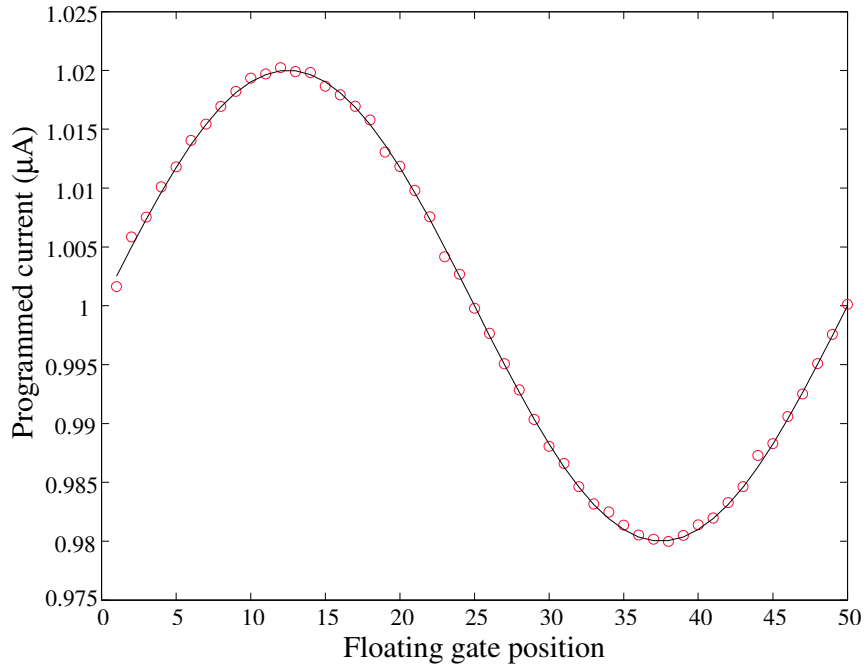


(a)

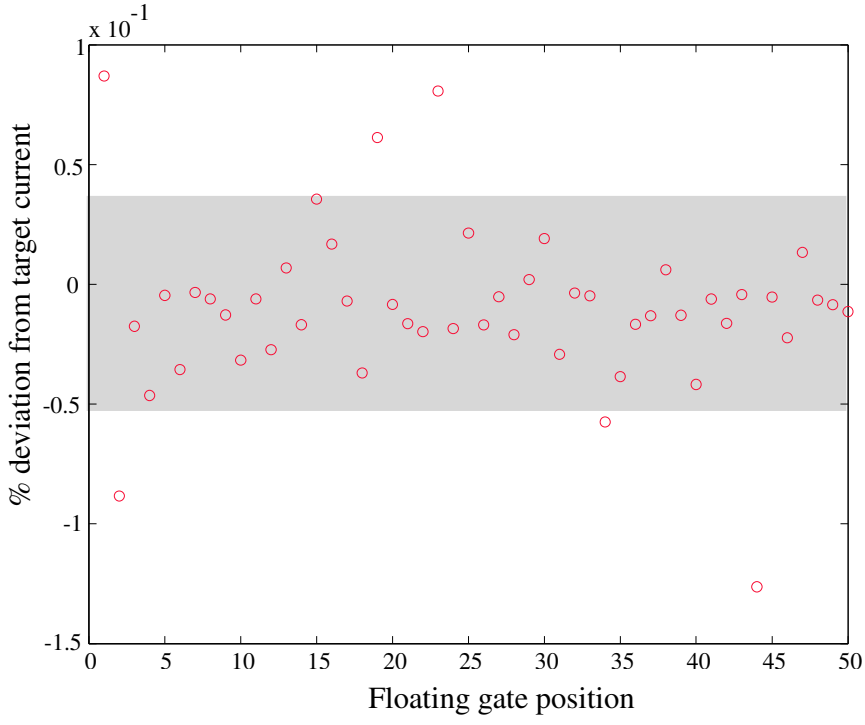


(b)

Figure 29: Subthreshold programmed values: Sine waves of different amplitudes were programmed on to the floating–gate transistors. The DC of the sine waves were 100nA. The bubbles indicate the stored values while the solid lines represent the ideal sine waves. The percentage deviation for each element was signal independent.



(a)



(b)

Figure 30: Above threshold programmed values: (a) A programmed sine wave of 40nA p-p that was programmed onto 50 floating gates. The DC of the sine wave was 1 μA. The bubbles indicate the stored values while the solid line represents the ideal sine wave. (b) Percentage error for the programmed sine wave.

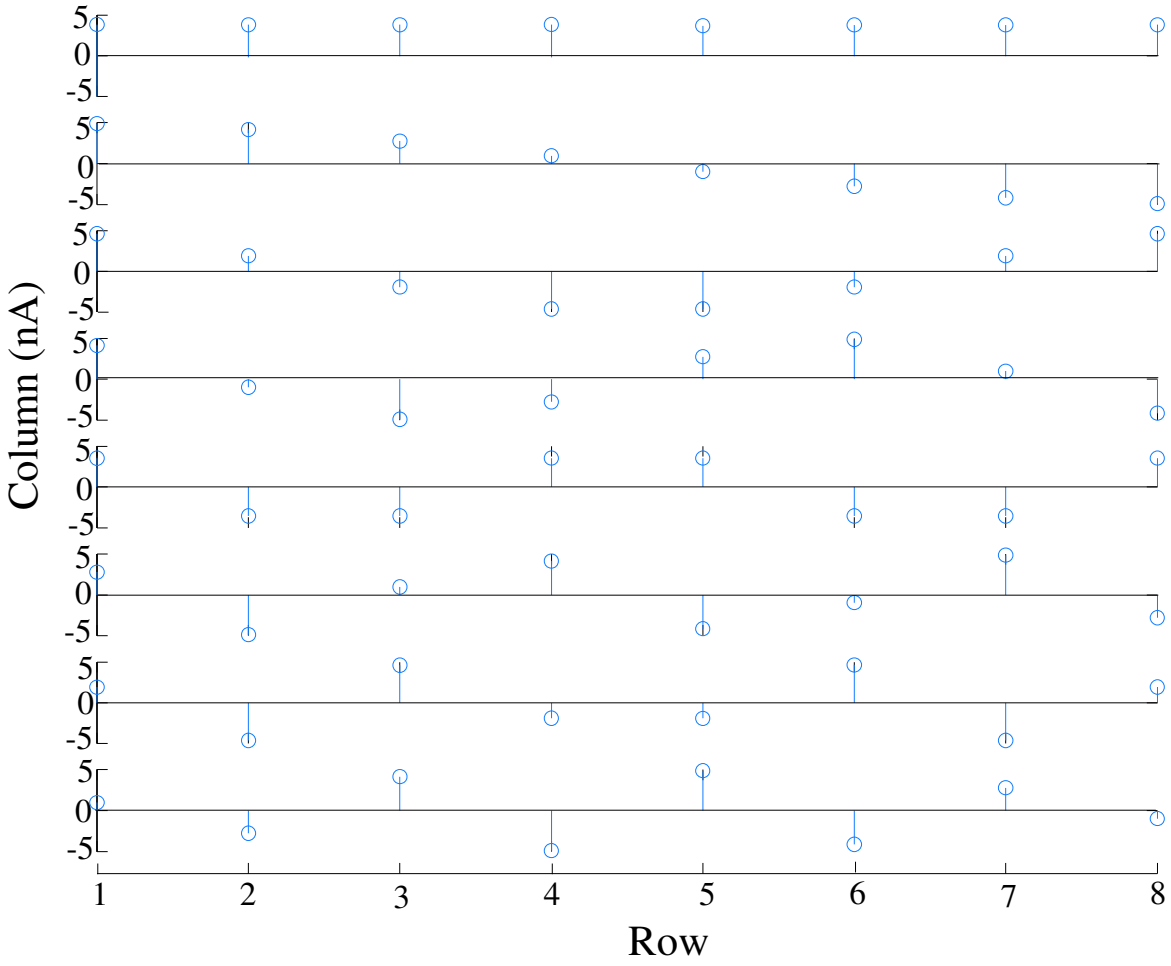


Figure 31: Programmed DCT values: A 8x8 DCT kernel was programmed onto a 8x8 array. The values were programmed around a DC of 10nA. The DC was subtracted for clarity of display. The maximum percentage deviation for the array was 0.07%.

pulses required to approach the target currents asymptotically were 4-6 pulses. It can be observed that the maximum error is less than 0.2%. This algorithm works over a wide range of currents. A 40nA p–p sine wave having a DC of 1μA was programmed using this algorithm as shown in the inset of Fig. 30. It also shows the error per element for this operation. The percentage error was $\pm$0.05%. The percentage deviation for each element was signal independent. As an application specific example we chose to program a 8x8 DCT kernel on the array. The programmed values are shown in Fig. 31. The maximum average deviation for the programmed values of 8x8 array was 0.07%.

This algorithm was also tested using other floating–gate application chips fabricated in

Table 1: Summary of performance for programming algorithm

Technology	0.25μm and 0.5μm N-well CMOS
Floating–gate Dim.(W/L)	6λ / 4λ
Array size	96×16
Maximum % error	$< \pm 0.2\%$
Pulse width	20μs
I-V type	Integrator
Global erase	Fowler-Nordheim tunnel
Programming mechanism	Hot–electron injection
Target range (current)	150pA to 1.5μA
Avg. no. of pulses for programming	7-8

0.5μm N–well CMOS processes. Table 1 summarizes the chip and algorithm performance. Figure 32 shows the micrograph of the chip that was fabricated to test the programming algorithm. Since the algorithm does not depend on the size of the array, large arrays can be programmed in parallel using this algorithm.

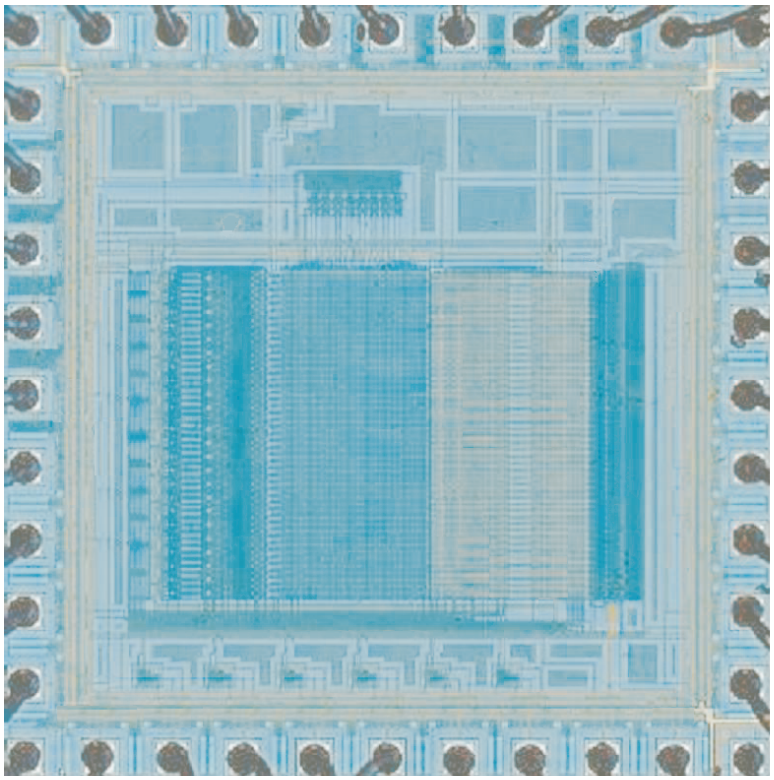


Figure 32: Die micrograph for programming chip: This chip was designed in 0.5μ N-well CMOS technology. It consists of an array of 96x16 floating–gate elements, on–chip I–Vs, and digital control for accurate programming.

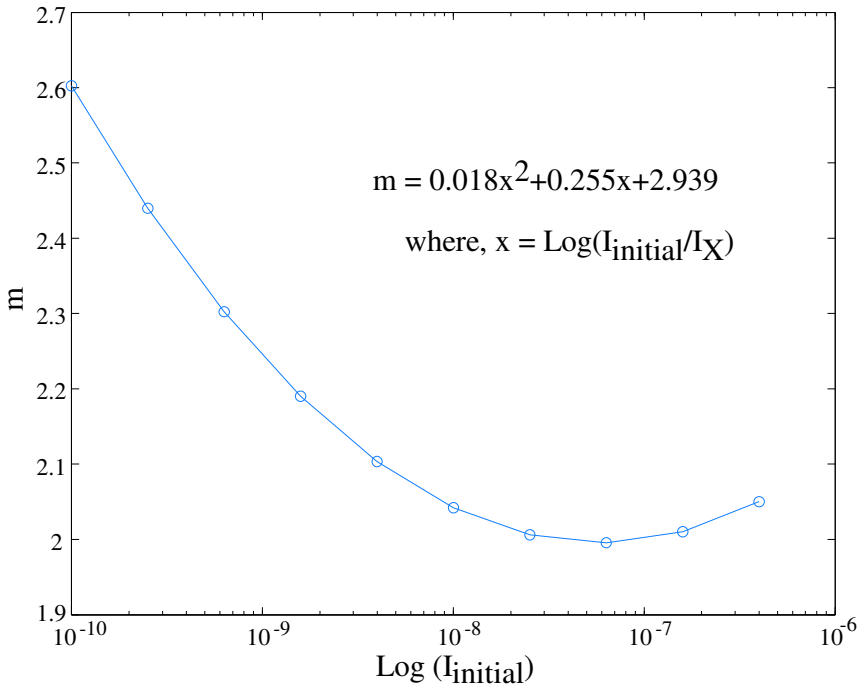


Figure 33: Variation of m with $I_{initial}$: The bubbles indicate slope of the data from Fig. 24. The line is a second order approximation of the same.

The usefulness of floating–gate memories will be determined by its ability to store the charge. Naturally occurring charge loss or charge gain has been described in [1]. Based upon extrapolations of these intrinsic charge loss mechanisms, it can be stated that a floating–gate memory can retain data for long periods at room temperature. It has been shown that the retention accuracy of a floating-gate memory is about 6-bits for 15 years at more than 125°C [1, 92]. Although these measurements were done using a different device from what we used in our experiments, we can expect a similar behavior.

2.7 Simplified model

The proposed method is computationally intense. It requires the storage of characterization parameters for a family of V_{DS}. A linear regression also has to be performed before the exact V_{DS} can be calculated.

The required computation can be simplified by observing the variation of m and f with $I_{initial}$ (Fig. 24). Figure 33 shows the variation of m with $I_{initial}$ while Fig. 34 shows the variation of f with $I_{initial}$. Both variations can be approximated by using second order

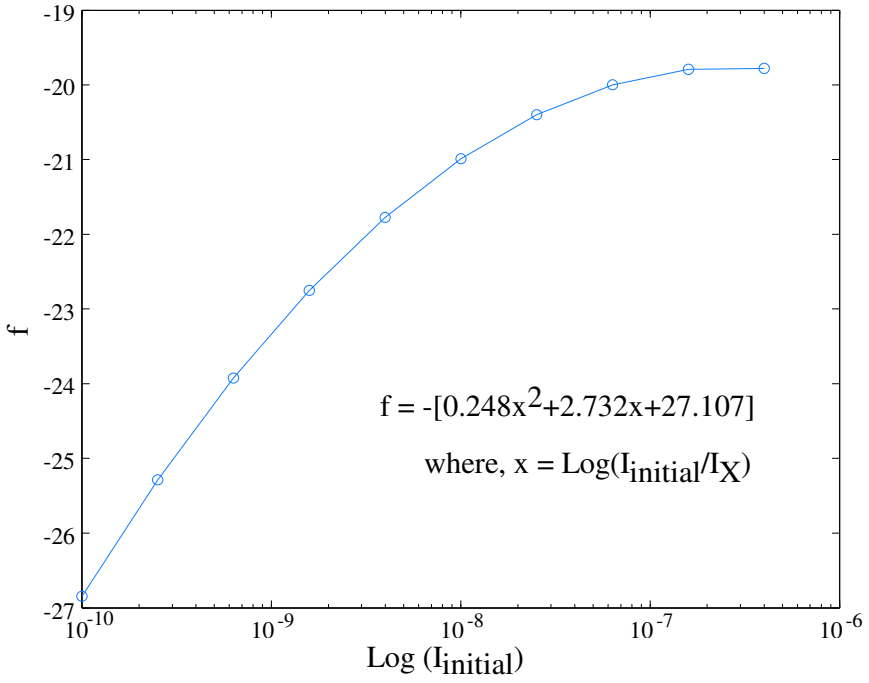


Figure 34: Variation of f with $I_{initial}$: The bubbles indicate intercept of the data from Fig. 24. The line is a second order approximation of the same.

relationships. The solid lines show the fit for each data set. Hence,

$$m = a_2 \left[\log\left(\frac{I_{initial}}{I_X}\right)\right]^2 + a_1 \log\left(\frac{I_{initial}}{I_X}\right) + a_0 \tag{3}$$

where a_0, a_1 and a_2 are regressed parameters with units of 1/Volts, and I_X is a scaling factor with units of Amperes.

Similarly,

$$f = b_2 \left[\log\left(\frac{I_{initial}}{I_X}\right)\right]^2 + b_1 \log\left(\frac{I_{initial}}{I_X}\right) + b_0 \tag{4}$$

where b_0, b_1 and b_2 are unitless regressed parameters, and I_0 is a scaling factor with units of Amperes. By substituting m and f from (2) with (3) and (4), and solving for V_{DS} we get:

$$V_{DS} = \frac{\log\left(\frac{\Delta I}{I_X}\right) - b_2 \left[\log\left(\frac{I_{initial}}{I_X}\right)\right]^2 + b_1 \log\left(\frac{I_{initial}}{I_X}\right) + b_0}{a_2 \left[\log\left(\frac{I_{initial}}{I_X}\right)\right]^2 + a_1 \log\left(\frac{I_{initial}}{I_X}\right) + a_0}, \tag{5}$$

where V_{DS} is a function of both, $I_{initial}$ and ΔI.

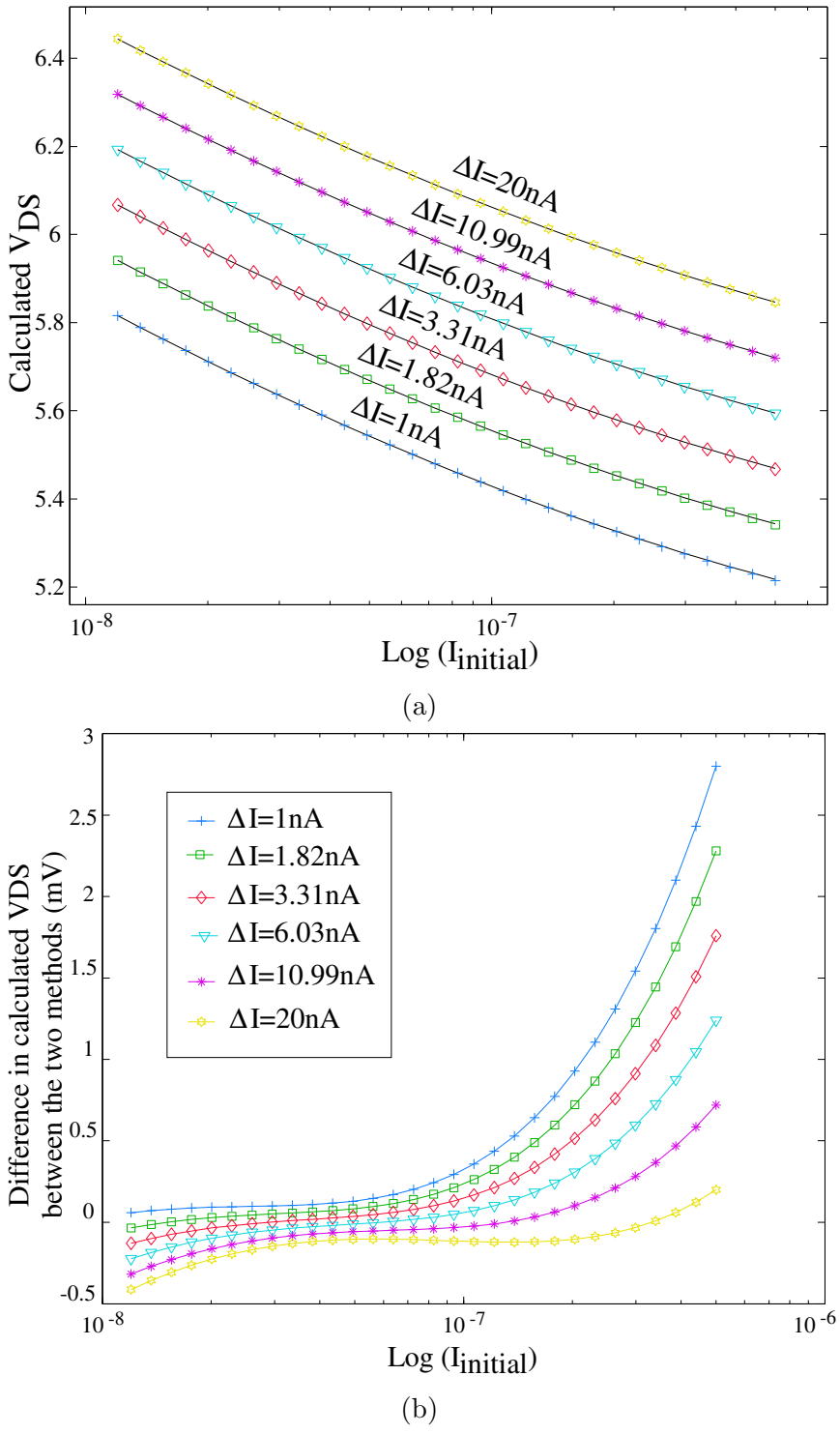


Figure 35: Comparison of methods: (a) The calculated V_{DS} is plotted for the two proposed methods. Symbols are for the first method while the solid line is for the simplified version. (b) Plot of deviation of calculated V_{DS} values using the two proposed methods.

Since this equation is a direct calculation of V_{DS}, the computational complexity has been reduced, as compared to the previous approach. Only six parameters (a_0, a_2, a_2, b_0, b_1, and b_2) need to be stored, and no regression has to be performed. Figure 35(a) shows the calculated V_{DS} using the two methods, while Fig. 35(b) shows the deviation of between V_{DS} calculated using the simplified model and the V_{DS} calculated using the first method. It can be seen that the error is very small and lesser than the DAC resolution which was used for programming. Experimental measurements showed that the programming accuracy of this simplified method was the same as the first method described.

2.8 Conclusion

This chapter presents a predictive algorithm that can be used to program large arrays of floating–gate elements at fast rates with 0.2% of accuracy over a wide range of target currents (over 3.5 decades). Experimental measurements and examples for different applications have been demonstrated. A simplified less computation intensive implementation has also been explored. Hot–electron injection is used for better control of programming over wide ranges. The algorithm was tested using large floating–gate arrays in 0.25μm and 0.5μm N-well CMOS processes.

CHAPTER III

BASIC TRANSFORM IMAGER PIXEL ELEMENT

3.1 Imager pixels

Pixels are the basic building blocks for any imager. There have been various topologies over the years with some being used just for reading an image while others for performing more complex computations with resultant reduction in fill–factor. Some of the common pixels are: logarithmic compression pixel, logarithmic compression pixel with feedback amplifier, buffered logarithmic pixel, adaptive photo–pixel, current amplifier pixel, passive pixel, and active pixel. Most of the pixels used today are variants of these photo circuits.

The matrix transform imager architecture (MATIA) pixel which can be used for performing matrix multiplications at the focal plane is introduces in this chapter. For performing matrix multiplications one needs to have the ability for element–by–element multiplication and also sum of products along a column. The MATIA pixel performs the element–by–element multiplication in the focal plane, while having a high fill–factor (46%). The output of this multiplication is drain currents. Since the processing is being carried out in current–mode, and the pixels along a column are connected together, the sum of products along the column is achieved by Kirchoff's current addition law.

This chapter consists of two sections. Section 3.2 introduces the MATIA pixel and discusses pixel tessellation issues. Section 3.3 discusses the various characterization that was carried out on the MATIA pixel like: dark currents, signal to noise, gain and offset mismatch across an array, linearity and harmonic distortion, correction of computation errors while using MATIA and bandwidth.

3.2 MATIA pixel

Each pixel is composed of a photodiode sensor element and an analog multiplier. Figure 36(a) shows that the circuit element for this multiplication is an nFET differential pair.

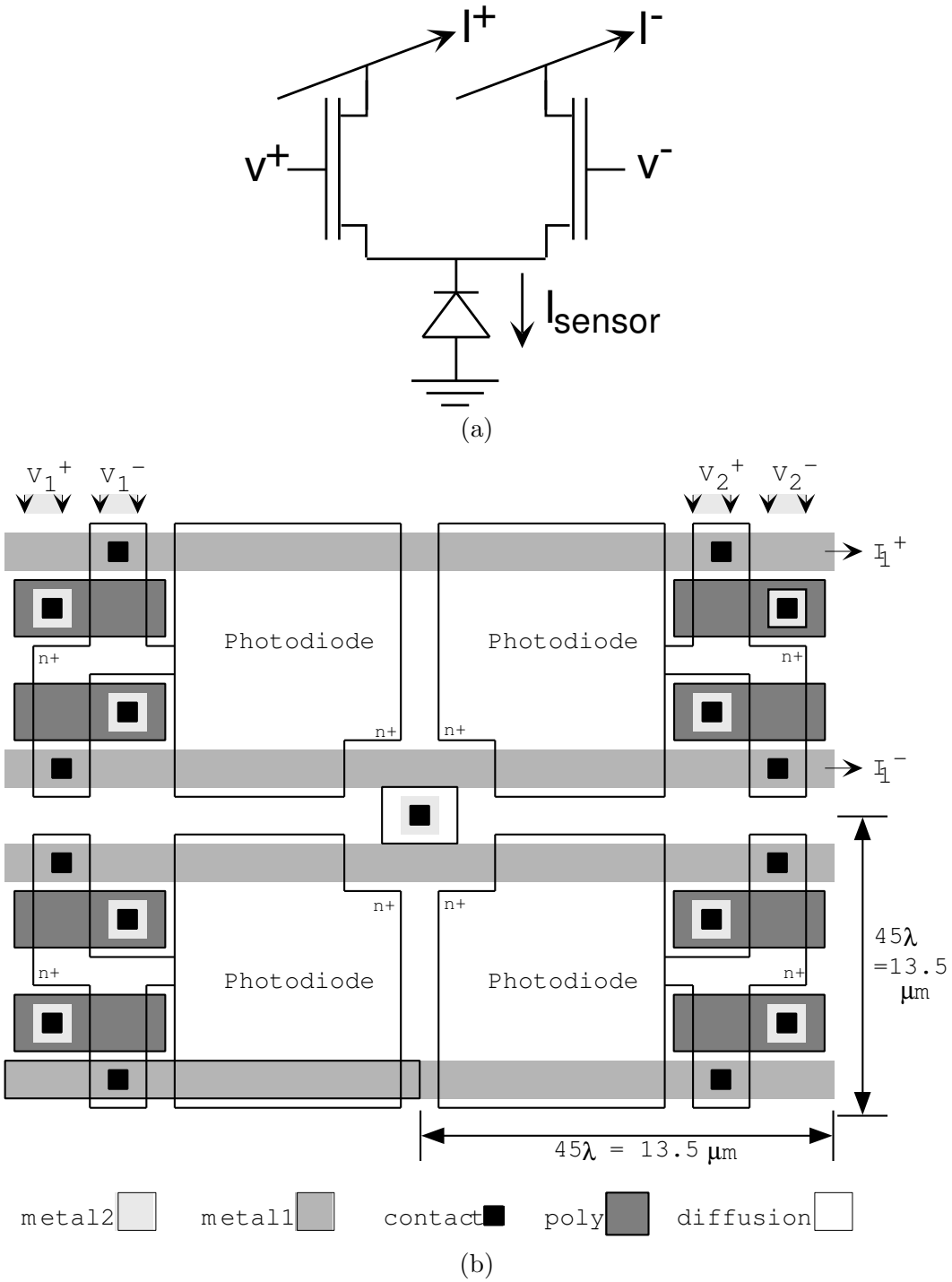


Figure 36: Transform imager pixel: (a) Basic Circuit: To multiply the transduced photodiode current by incoming basis functions, we use a differential pair to modulate a fraction of the sensor current through the transistors. (b) Layout of a 2×2 pixel array. The pixel has a rectangular tessellation and has a fill–factor of 46%. The dimensions given are those used in fabricating this element in a double poly 0.5μm CMOS process.

Figure 36(b)a shows the layout for a 2x2 pixel block. The pixel has a size of 13.5μm x13.5μm. For the differential pair operating with subthreshold bias currents (which should always be the case due to the low–level image sensor currents), we can express the differential output current as,

$$I^+ - I^- = I_{sensor} \tanh\left(\frac{\kappa(V^+ - V^-)}{2U_T}\right) \tag{6}$$

where κ is the gate coupling efficiency into the transistor surface potential (typically 0.6 - 0.8), and U_T is kT/q [87]. When this circuit is in its linear range, that is when $V^+ - V^-$ is less than $2U_T/\kappa$, we get

$$I^+ - I^- = I_{sensor}\left(\frac{\kappa(V^+ - V^-)}{2U_T}\right) \tag{7}$$

therefore the differential output current is a linear product of the sensor current and the applied differential voltage.

The experimental data in Fig. 37 shows that linear multiplication takes place within the linear range. A single pixel would result in 300pA current levels from typical room fluorescent lights at roughly 2 meters from the imager without a lens to focus the light. This pixel could include more advanced image sensor elements or circuits with a corresponding modification to the resulting fill factor. Additionally, each pixel could be directly read out by this technique, since a column scan is equivalent to multiplication by a digital value moving by one position for each step ($\tanh(x) \approx 1$ or -1 for large values of x).

3.2.1 Pixel Structure and Tessellation

An imager maps light intensity from a three dimensional space into a two dimensional image in the focal plane. For any bandlimited signal there is an infinite number of possible ways to sample the signal. In 1–D systems regular sampling is described by a single parameter, the sample period T. In general, the sampling pattern is determined by two independent vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. In 1–D systems, the sample points are defined as $t = nT$ whereas in 2–D systems, the sample points are defined as $\mathbf{t} = n_1\mathbf{v}_1 + n_2\mathbf{v}_2$. For convenience the sampling vectors are usually combined into a matrix $\mathbf{V} = [\mathbf{v}_1|\mathbf{v}_1]$ [33].

The issues that need to be considered while choosing the tessellation structures of vision chips are as follows:

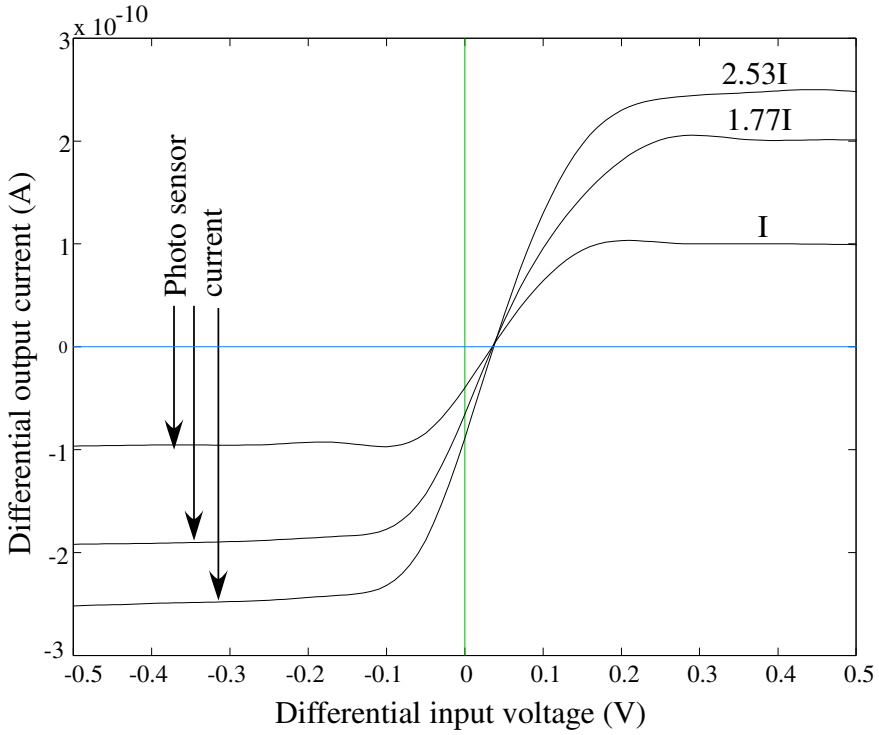


Figure 37: Operation of pixel: A DC characterization of the pixel was carried out using various intensity of input illumination (I, 1.77I, 2.53I). This corroborates the theoretical transfer function.

- Neighborhood distance
- Layout complexity
- Data manipulation
- Biological plausibility

Among the numerous possibilities only two sampling strategies are common – rectangular and hexagonal sampling. Hexagonal sampling is more symmetric with respect to neighborhood connectivity and distance maps, hence produces the same equidistant maps for different metrics (such as Euclidean, city block, or chess board) between neighboring pixels. Hexagonal tessellation of the image plane is also of great interest because it more similar to the sampling used in most biological systems. In addition, for band–limited signals, hexagonal sampling in two dimensions results in a 13.4% efficiency gain in the number of sample points required. However, the large number of interconnects required for

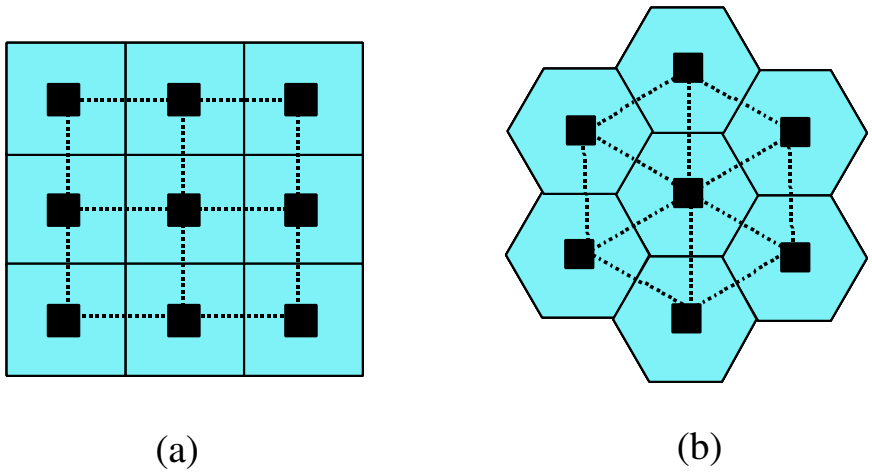


Figure 38: Pixel tessellation: Rectangular and hexagonal sampling are the most common types of sampling used in imagers. The choice of the type of sampling used depends on the application and the complexity of the design.

a hexagonal pixel array along with the associated layout complexity makes the hexagonal pixel configuration less attractive as compared to the rectangular configuration. Due to its simplicity, a rectangular sampling grid is the preferred option in this architecture. A rectangular tessellation provides a more convenient representation of a 2-D image. Furthermore, the savings in area realized by using a rectangular structure in place of a hexagonal structure allow us to have a denser imager with higher fill–factor and more pixels per unit chip area.

The only potential problem that remains when using rectangular sampling to model a hexagonally sampled system is the implementation of various operations. For linear operations this problem can be overcome by resampling the band–limited impulse response of the desired filter operation [33]. Since our sampling grid is more dense than could be achieved by using a hexagonal tessellation of the imager, nothing is lost relative to a hexagonal architecture.

In addition to chip area economy arguments, fill–factor has signal processing implications as well. The finite photodetector size acts as a simple spatial averaging window. This appears as a *sinc* function $(\sin(x)/x)$ in the spatial frequency domain and causes aliasing of the high spatial frequency contents of the image. Increasing photo–receptor area decreases this aliasing effect. Any aliasing that is not decreased by increased fill–factor must be

reduced by applying optical anti-aliasing filters (*e.g. intentional blurring*) [91].

3.3 Characterization of MATIA pixel

In practice, the elements will not be perfect multipliers and will not be exactly identical to the other elements. The MATIA pixels were characterized for dark currents, frequency response, offset errors, linearity mismatch, gain and kappa mismatch. For this architecture and algorithm, as long as offsets and linear range are bounded, the errors are set by gain errors. This error occurs primarily due to kappa mismatch. The MATIA pixel has square edges and the various mismatches can be reduced by reducing the edges in the pixel layout. This section contains results from those characterization experiments.

3.3.1 Dark current

Dark current is due to charge generation independent of any input signal. The total dark current density can be expressed as [17]:

$$J_d = qn_i[\frac{n_i L_p}{n_o \tau_p} + \frac{W}{2\tau_o} + \frac{v_r}{2}] + q\eta\phi_B + J_{tunnel} \tag{8}$$

where n_i is the intrinsic carrier concentration, n_o is the equilibrium concentration, L_p is the diffusion length, τ_p is the hole lifetime, W is the depletion region width, q is the electronic charge, η is the quantum efficiency, ϕ_B is the background photon flux density, v_r is the maximum surface recombination velocity, and J_{tunnel} is the tunneling current.

The first term is due to the minority carrier diffusion current generated in the bulk region; the second term is the drift current within depletion; the third term is the surface recombination current due to generation-recombination processes by means of interface states; the forth term is the background current due to photo–generated carriers; and the fifth term is due to electrons tunneling from the valence bond to the conduction band [17]. Tunneling is important in narrow–bandgap materials. In these materials, under sufficient bias, electrons can tunnel out of the valence band into the conduction band, thereby leaving holes behind in the valance band and producing electrons in the conduction band.

The MATIA pixel was characterized for dark current by measuring the current under

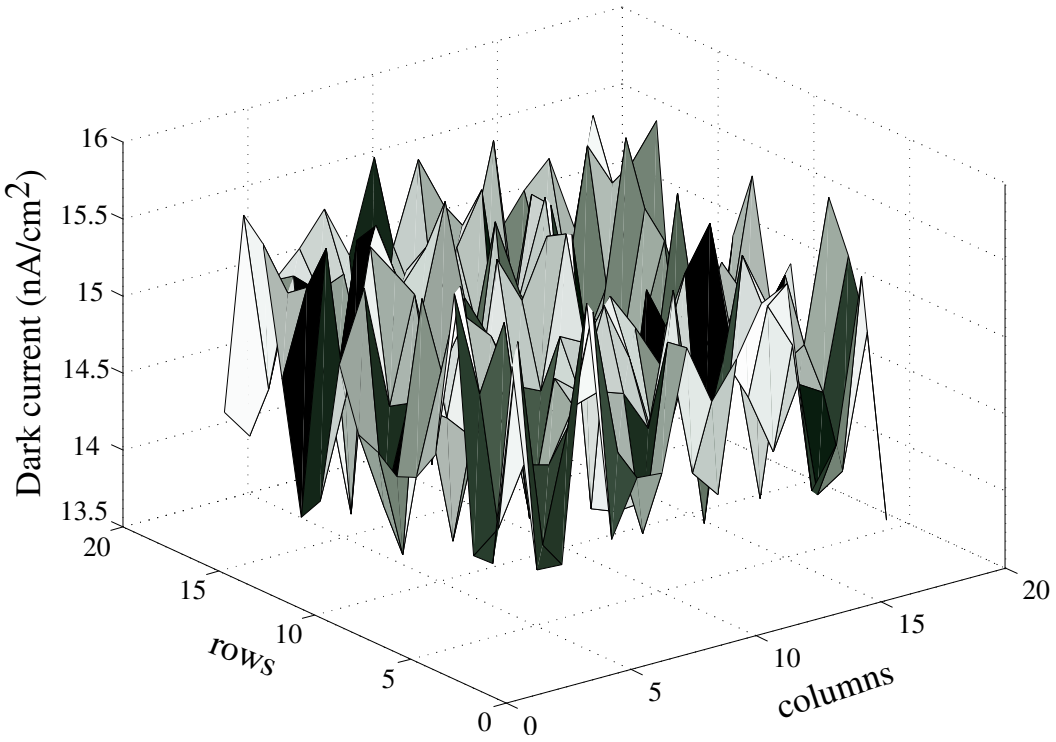


Figure 39: Dark current distribution: This figure shows the dark current distribution in a 16×16 block taken from the middle of the imager. The average dark current was measured to be 14.9 nA/cm^2

reset condition with no chip illumination. Figure 39 shows the distribution of dark currents from a 16×16 block of a larger imager. The average dark current was measured to be 14.9 nA/cm^2. Edge effects are not observed in this case as this block was from the middle of the imager. The variations have been observed to follow no trends and are random in nature. These measurements were taken under no illumination conditions with the MATIA configured to read an image. Conventional dark current methods would give a lower value than that measured here, as we have extra parasitics in the signal path when using our method of measuring dark currents. In applications where very high performance (and therefore, nearly zero offsets) is required, one could use floating–gate tuning techniques [25], with the accompanying decreases in fill factor, or subtract these offsets from a bank of floating-gate elements.

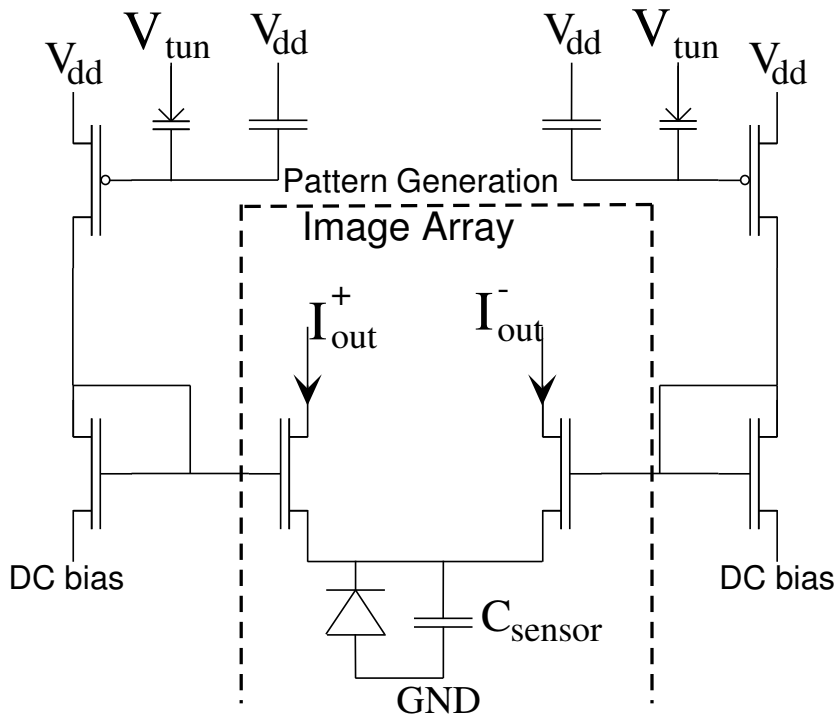


Figure 40: **Effective analytical circuit for estimating noise and computation speed**: Part of this circuit is in the pixel array, and part of the circuit is in the basis generating circuits. The output current lines need to be appropriately terminated, typically with active feedback to eliminate the resulting line capacitance.

3.3.2 Signal–to–noise issues

Since the pixel currents are fairly low (subthreshold), thermal noise contributes to most of the transistor noise. Thermal noise is modelled as

$$\hat{I}^2 = 2qI\Delta f \tag{9}$$

$$\frac{\hat{I}^2}{I^2} = \frac{2q}{I}\Delta f \tag{10}$$

where I is the current level; larger current results in larger noise power, but smaller percentage of noise. The bandwidth (Δf) is the approximate highest frequency (i. e. the fastest generated signal) of the basis generator. For a 1 million pixel array and $\Delta f = 60$kHz the relative noise level is 0.14 for a 1pA bias current through a single transistor.

Due to the low currents, 1/f noise only becomes noticeable at low frequencies (e. g. 10Hz). Noise generated at frequencies less than the frame rate will be eliminated from the final computation, so the low 1/f noise will not affect these circuits. This property is similar

to the computation in correlated double-sampling techniques. Therefore, we only need to address thermal noise generated from the sensor circuits.

It can be shown that this circuit has effectively three differential pairs worth of noise (1.5 transistors worth of noise–assuming noise power splits evenly between differential and common mode) at the bias current determined by the light incident on the pixel sensor. Figure 40 shows the effective circuit to look at this noise analysis. The noise comes from two sources: the differential pair transistors on the photodiode contributes one differential pair worth of noise while the basis generation structures contribute two differential pairs worth of noise at the sensor's bias current. For very small signals, the system looks like a current mirror with different transconductances (the gain = g_{m2}/g_{m1}), resulting in two differential pairs worth of noise (two because of no common-mode rejection for this circuit component). The noise from the photodiode gets modulated in–band by the incoming signals, and therefore, is negligible in practice.

Noise power is equal to, $2qI_{signal} + 2\left(g_{m2}/g_{m1}\right)2qI_{bias}$. If M_1 is operating in the sub-threshold region, then $g_{m1}/I_{bias} = \kappa/U_T$. For noise, $2qI_{signal} + 2\left(g_{m2}/g_{m1}\right)2qI_{bias} \approx 2qI_{signal}(1+2) = 3(2qI_{signal})$.

The next issue is the resulting noise of a single pixel or group of pixels in an array. Since each noise source is independent of the other noise sources, the noise power of each source increases linearly with the number of sources (N). Therefore, the noise relative to the signal from a single pixel is

$$\hat{I}^2 = 2qIN\Delta f \tag{11}$$

$$\frac{\hat{I}^2}{I^2} = \frac{2qN}{I}\Delta f \tag{12}$$

For the 1 million pixel example above, the relative noise level for the entire pixel sensor is 6.73 (-16.6dB) for a 1pA bias current and 0.673 (3.43dB) for 100pA bias current. For a completely correlated feature, which means all 1K elements contribute to a large output signal, we get a relative noise level of 0.0066 (43.6 dB). Therefore, for this imager setup, either higher illumination or more coherent features (features selected by the basis generator) will result in increased higher SNR. This SNR value is similar to the SNR if each pixel

was measured at the 60Hz frame rate; therefore, correlated features have the same SNR as reading the pixel array, but uncorrelated pixels will have very low SNR.

3.3.3 Gain, offset mismatch

The pixel operates in the subthreshold region as the photo currents are very small (tens of nA). For long channel the subthreshold drain current can be written as [17]:

$$I_D = \mu \frac{C_{ox}}{\kappa} \frac{W}{L} U_T^2 (exp \frac{\kappa(V_{GS} - V_{TH})}{U_T})(1 - exp \frac{-V_{DS}}{U_T}) \tag{13}$$

where C_{ox} denotes the oxide capacitance, $U_T = kT/q$, $\kappa = (C_{ox})/(C_{ox} + C_d + C_{FS})$ is the gate coupling efficiency into the transistor surface potential (typically 0.6 - 0.8), $C_d = \sqrt{\epsilon_{si} q N_{sub}/(4\phi_B)}$ denotes the capacitance of the depletion region under the gate area, and C_{FS} is the capacitance due to fast surface states. In saturation ($V_{DS} \geq 4U_T$) the above expression reduces to:

$$I_D = \mu \frac{C_{ox}}{\kappa} \frac{W}{L} U_T^2 (exp \frac{\kappa(V_{GS} - V_{TH})}{U_T}) \tag{14}$$

Hence solving for V_{GS},

$$V_{GS} = \frac{U_T}{\kappa} ln(\frac{I_D \kappa}{\mu C_{ox} \frac{W}{L} U_T^2}) + V_{TH} \tag{15}$$

To calculate the input referred voltage for two devices in Figure 41, the mismatches are incorporated as $V_{TH1} = V_{TH}$, $V_{TH2} = V_{TH} + \Delta V_{TH}$, $(W/L)_1 = (W/L)$, $(W/L)_2 = (W/L) + \Delta(W/L)$, $I_{D1} = I_D$, $I_{D2} = I_D + \Delta I_D$. For simplicity, the variations in the other terms are neglected.

Since $V_{OS,in} = V_{GS1} - V_{GS2}$, we have

$$V_{OS,in} = \frac{U_T}{\kappa} [ln\{\frac{I_D}{I_D + \Delta I_D} \frac{(W/L) + \Delta(W/L)}{(W/L)}\}] - \Delta V_{TH} \tag{16}$$

$$V_{OS,in} = \frac{U_T}{\kappa} [ln\{\{1 + \frac{\Delta I_D}{I_D}\} + ln\{1 + \frac{\Delta(W/L)}{(W/L)}\}] - \Delta V_{TH} \tag{17}$$

Since, $ln(1 + x) \approx x$ for small values of x, we have

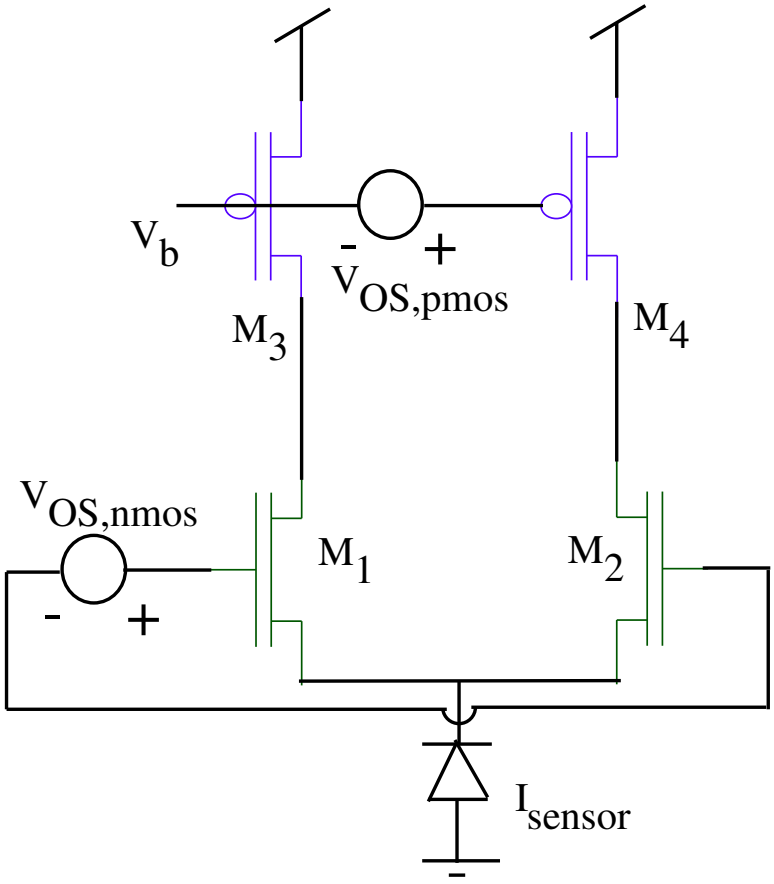


Figure 41: Schematic for offset calculations : The effective circuit for input referred offset voltage calculations for the basic pixel structure along with readout transistors

$$V_{OS,in} = \frac{U_T}{\kappa}[-\frac{\Delta I_D}{I_D} + \frac{\Delta(W/L)}{(W/L)}] - \Delta V_{TH} \tag{18}$$

This equation reveals the dependence of $V_{OS,in}$ on device mismatches and bias conditions. Just like in the above threshold case the threshold voltage mismatch is directly referred to the input. Hence the input referred voltage can be reduced by increasing the aspect ratio of the input transistors. Also the variation of threshold voltage can be approximated as,

$$\Delta V_{TH} = \frac{A_{ox}}{\sqrt{WL}} mV \tag{19}$$

where A_{ox} is a proportionality factor determined from measurements. A_{ox} has been observed to scale down with gate–oxide thickness. Therefore, ΔV_{TH} can be reduced by increasing

the area of the input transistors. Since the channel capacitance is proportional to WLC_{ox}, we note that ΔV_{TH}, and the channel capacitance bear a trade–off. Also in a pixel fill–factor places a constraint on the size of the transistors to be used in the pixel.

The basic readout circuitry is shown in Figure 41. The total input referred voltage for the circuit can be written as:

$$V_{OS,in,TOT} = V_{OS,pmos}\frac{g_{m,pmos}}{g_{m,nmos}} + V_{OS,nmos} \tag{20}$$

Therefore, for low input offset voltage $g_{m,pmos}$ should be made smaller than $g_{m,nmos}$. Since in subthreshold region for our configuration $g_{m,nmos} = g_{m,pmos}$, as they are both proportional to current flowing through them (same in this case), the total input referred voltage is as follows:

$$V_{OS,in,TOT} = V_{OS,pmos} + V_{OS,nmos} \tag{21}$$

It should be noted that in this case $V_{OS,nmos}$ is the dominant term and is much larger than $V_{OS,pmos}$, as the PMOS acts as a switch and the on–resistance will match as they have been laid out next to each other. Although the values across a large array may not match that accurately when comparisons are made between switches from different positions in the array.

The offset voltages for a small array of pixels are shown in Figure 42. The mean and the standard deviation are 8.9mV and 6.7mV respectively. Offset errors are primarily due to offsets in the differential pair transistors. As long as the modulation signal is roughly within the linear range of the differential amplifier, we can eliminate offsets by eliminating the low frequency signal (less than the frame rate) from the result, because there is no signal at these low frequencies (we are modulating the pixels) except for the effect of offsets. Pixels with these large offsets will result in significant image distortion at these points. We found that most of the offsets were within 10mV of the other elements along the column. We can account for average column offsets by appropriately programming the input basis functions

Gain error is primarily due to κ mismatch in the differential pair transistors. Typically κ matches fairly well for transistors with similar currents and for similar source voltages.

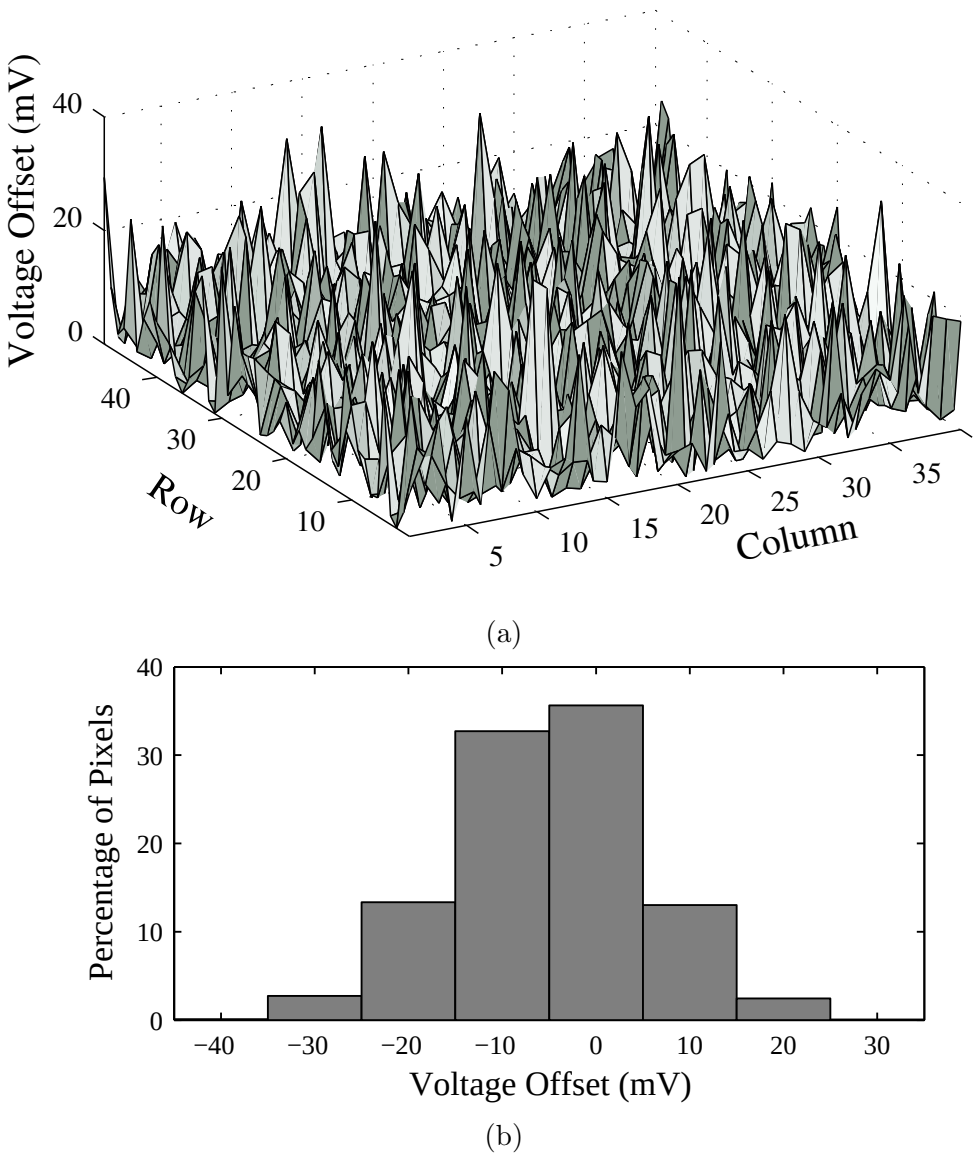


Figure 42: Variations of voltage offsets: The variations of voltage offsets for a 48x40 array is shown above. The mean and the standard deviation are 8.9mV and 6.7mV respectively.

The variation of gain for a small array of pixels are shown in Fig. 43(a). The mean and the standard deviation are 2.12nA/V and 0.0336nA/V respectively. The histogram of gain is plotted in Fig. 43(b). Similarly the variation of κ is determined from the gain plots. Figure 44 shows the variation of κ with position. The mean and the standard deviation are 0.7149 and 0.0072 respectively.

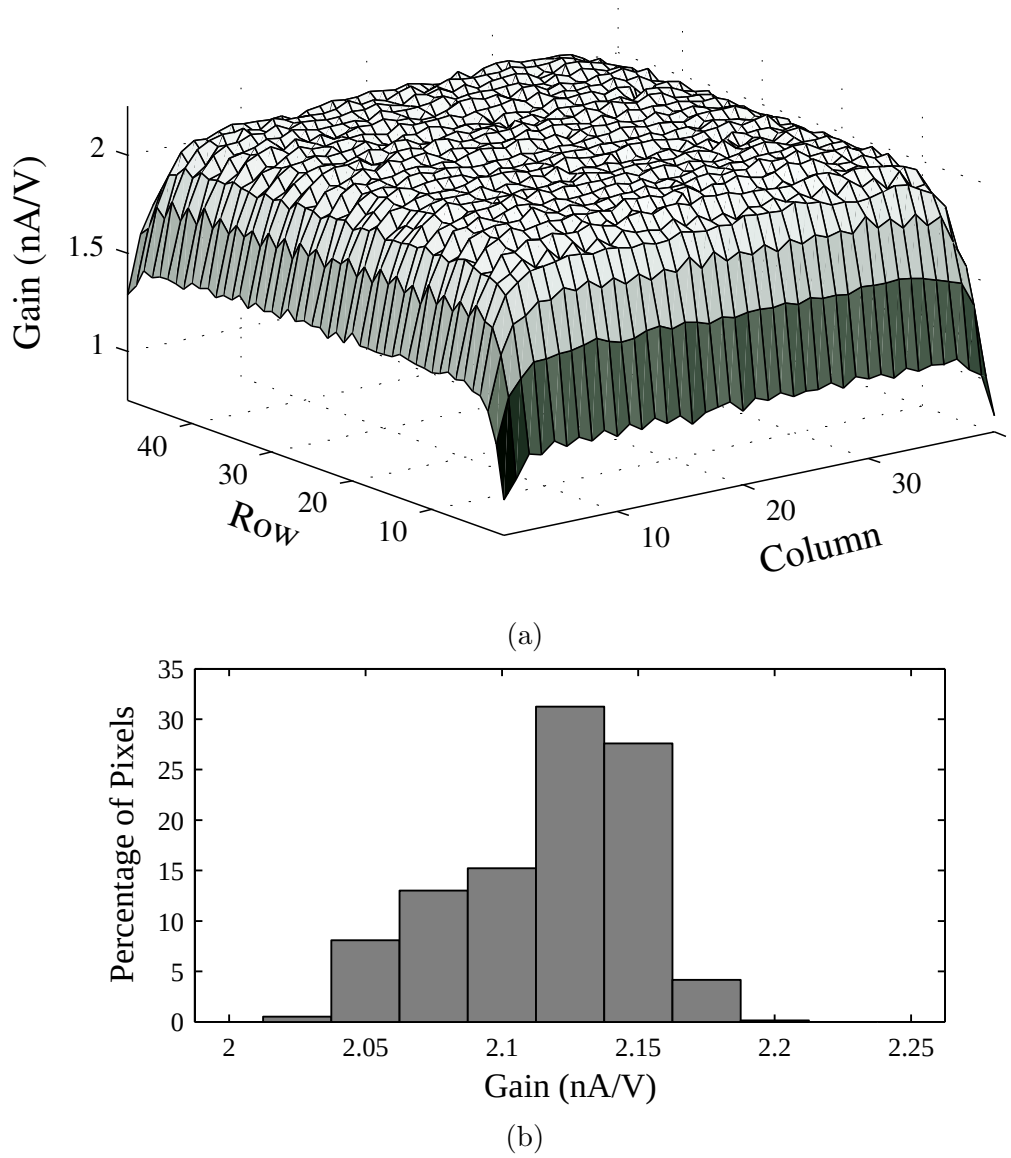


Figure 43: Gain mismatch across the array: The (a) spatial variations, (b) histogram of gain mismatches for a 48x40 array is shown above. The mean and the standard deviation are 2.12nA/V and 0.0336nA/V, respectively. This mismatch is primarily due to the mismatch of κ.

3.3.4 Linearity and Harmonic distortion of the pixel

Since the pixel is used for multiplication linearity of the pixel is important. The linearity of the pixel is extracted from the DC sweeps. Figure 45 shows the spatial variation and the histogram of range of linear multiplication with position. The mean and the standard deviation are 54.4mV and 4.3mV respectively. Harmonic distortion can be analyzed using the transfer function for the pixel.

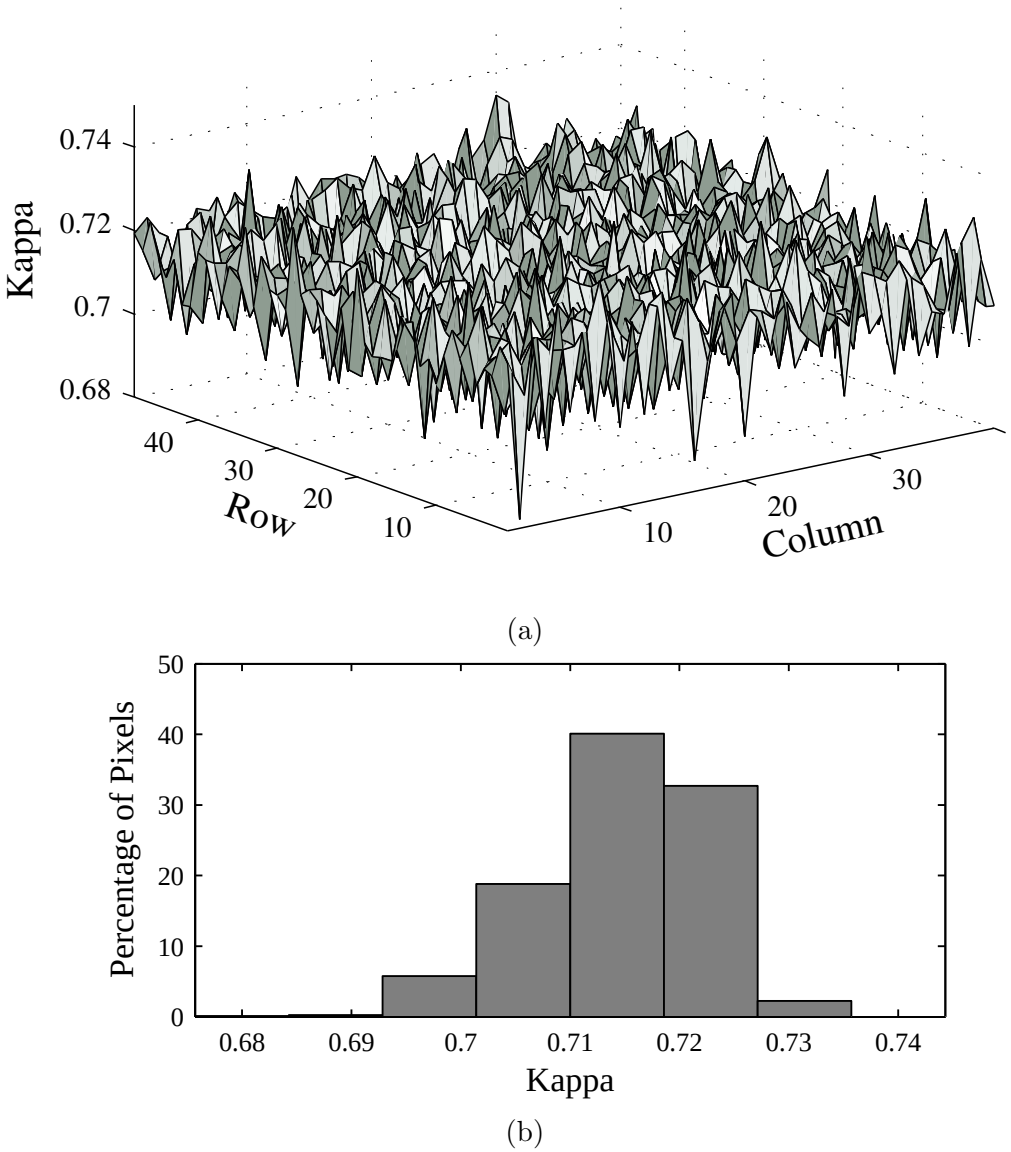


Figure 44: Variation of kappa: The variations of Kappa for a 48x40 array is shown above. The mean and the standard deviation are 0.7149 and 0.0072 respectively. This mismatch is primarily due to the mismatch of the gate coupling efficiency into the transistor surface potential.

The transfer function of the pixel can be written as:

$$\Delta I = I_{sensor} \tanh\left(\frac{\kappa(\Delta V)}{2U_T}\right) \tag{22}$$

if $m = \frac{\kappa(\Delta V)}{2U_T}$ then,

$$\Delta I = I_{sensor} \tanh m = I_{sensor} \frac{sinh(m)}{cosh(m)} \tag{23}$$

now,

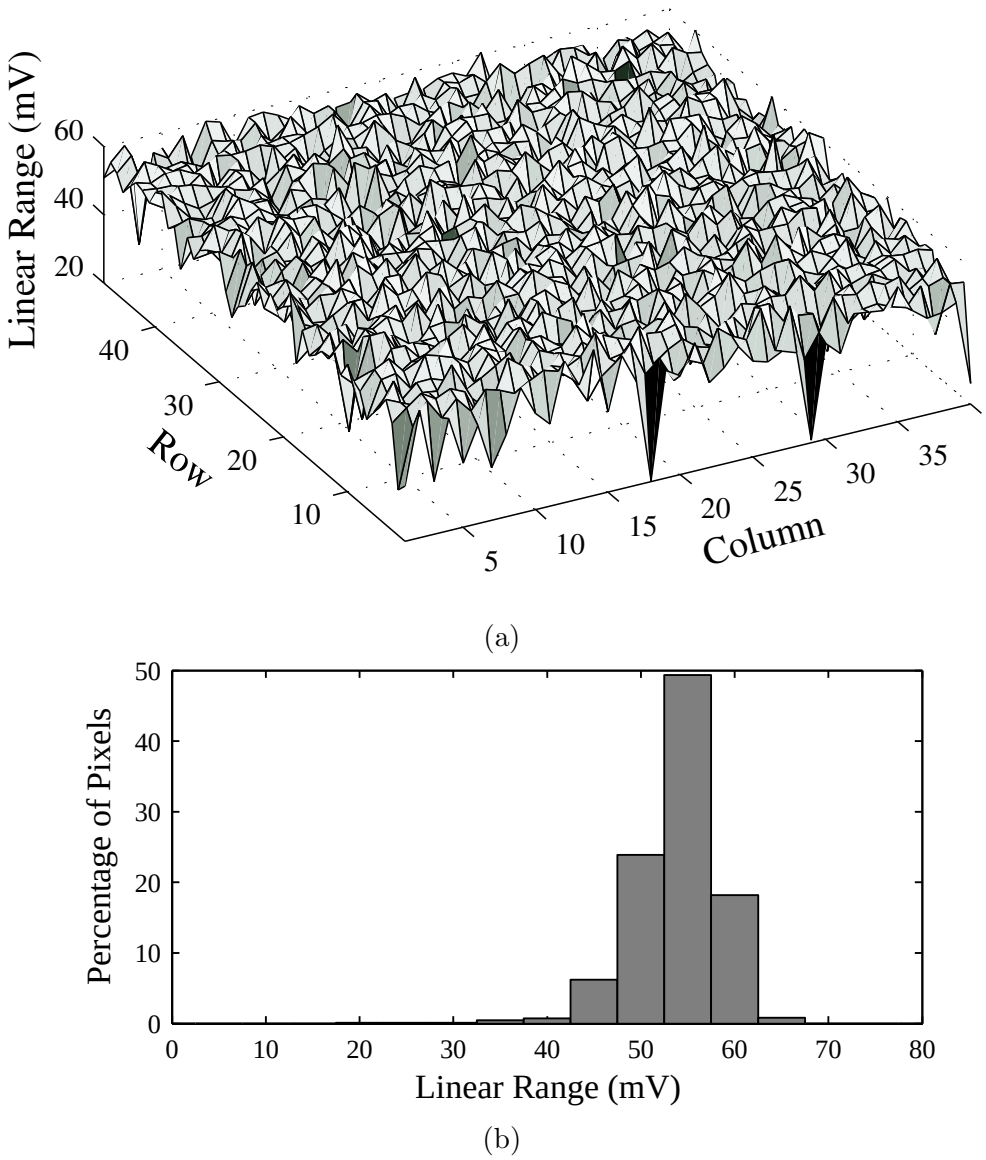


Figure 45: Variations of linearity: The variations of linearity for a 48x40 array is shown above. The mean and the standard deviation are 54.4mV and 4.3mV respectively.

$$sinh(m) = m + \frac{m^3}{3!} + \frac{m^5}{5!} + \frac{m^7}{7!} \tag{24}$$

and,

$$cosh(m) = 1 + \frac{m^2}{2!} + \frac{m^4}{4!} + \frac{m^6}{6!} \tag{25}$$

Therefore substituting these in Eq. 22,

$$\Delta I = I_{sensor}[\frac{m + \frac{m^3}{3!} + \cdots}{1 + \frac{m^2}{2!} + \cdots}] \tag{26}$$

for $m \ll 1$,

$$\Delta I = I_{sensor}[(m + \frac{m^3}{3!} + \cdots)(1 - \frac{m^2}{2!} + \cdots)] \tag{27}$$

$$\Delta I = I_{sensor}[m - \frac{m^3}{3} + \cdots] \tag{28}$$

substituting for m, and $\Delta V = V_s cos(wt)$,

$$\Delta I = I_{sensor}[\frac{\kappa}{2U_T} V_s cos(wt) - (\frac{\kappa}{2U_T})^3 \frac{(V_s cos(wt))^3}{3} + \cdots] \tag{29}$$

since $cos^3(wt) = [3cos(wt) + cos(3wt)]/4$, we obtain the following after substitution and rearranging,

$$\Delta I = I_{sensor} \frac{\kappa}{2U_T} [V_s \{1 - (\frac{\kappa}{2U_T})^2 \frac{V_s^2}{4}\} cos(wt) - (\frac{\kappa}{2U_T})^2 \frac{V_s^3}{12} cos(3wt) + \cdots] \tag{30}$$

As expected from a differential circuit the even order harmonics are not present. From Eq. 30,

$$\frac{A_{HD3}}{A_F} = (\frac{\kappa}{2U_T})^2 \frac{V_s^2}{12[1 - (\frac{\kappa}{2U_T})^2 \frac{V_s^2}{4}]} \tag{31}$$

If $[1 - (\frac{\kappa}{2U_T})^2 \frac{V^2}{4}] \ll 1$ this simplifies to:

$$\frac{A_{HD3}}{A_F} \approx (\frac{\kappa}{2U_T})^2 \frac{V_s^2}{12} \tag{32}$$

It can be observed that the harmonics are independent of the photodiode current and depends on the input signal amplitude. Similarly the other odd order harmonics can be found out. The total harmonic distortion can be figured out from the expression:

$$THD = \frac{[A_{HD3}^2 + A_{HD5}^2 + \cdots + A_{HD(2n+1)}^2]^{1/2}}{A_F} \tag{33}$$

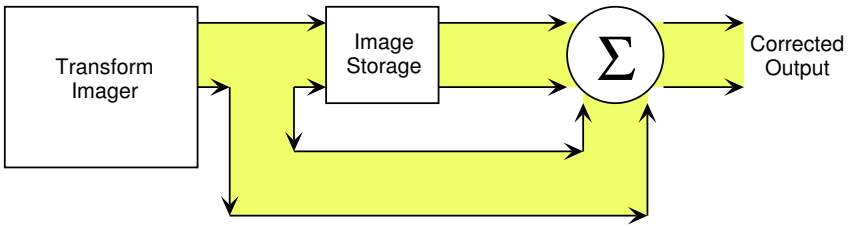


Figure 46: Correction for imager errors: Imager architecture for taking image differences; we need a separate array to store one frame. An array of floating-gate devices (similar to the basis generation array) would implement *image storage* for eliminating nearly constant images such as offset errors from dark currents, or constant background images.

Harmonic distortion effectively results in the spreading of modulation energy to other pixels. This spreading is independent of the sensor signals since the modulation signal stays at the same amplitude. The modulation signals can be modified to account for this *spreading* such that the transform is effectively free of this signal spreading.

3.3.5 Correction of computation errors in MATIA

We focus on multiplication errors because addition of currents by KCL is an ideal computation. Another source of error comes from the dark currents, which are typically in the femto–ampere range and therefore, are important for pixels operating in low–light levels. We can use floating-gate elements to eliminate them, as shown in Fig. 46. An array of floating-gate devices (similar to the basis generation array) would implement *image storage* for eliminating nearly constant images such as offset errors from dark currents, or constant background images. Currents can be scaled, and typically the current from a transform imager will be scaled as well; therefore, removing dark currents, which are typically in femto–ampere range, would be subtracted with a current in the high pico–ampere range. An array of sample–and–hold elements would implement *image storage* for temporal filtering and temporal derivatives associated with motion. This technique can be generalized for a wide range of temporal filters; the number of temporal delays proportionally increases the image storage. The advantage of subtracting a fixed image is that we get higher system density, since we don't need to integrate the two core cells into a single element with the supporting control logic. Also, any floating-gate elements are removed from potential UV light, therefore reducing any floating-gate charge drift issues.

One can modify the modulation signals to account for this *spreading* such that the transform is effectively free of this signal spreading. To analyze this problem, we decompose all modulated signals, $x(t)$, into a finite Fourier series because the signals repeat for each frame, and the signals have a maximum frequency by the clock rate of the basis generator. We write the Fourier series as

$$x_k(t) = \sum_{\ell=-N}^{N} a_{k\ell} e^{jw_{frame}\ell} \tag{34}$$

where a_{kl} is the ℓ^{th} coefficient for the kth signal, and w_{frame} is 2π times the frequency of the frame rate. Note $a_{k0} = 0$ because there is no DC signal component. In matrix form $\mathbf{x}(t) = \mathbf{A}\mathbf{f}(t)$ where $f_\ell(t) = e^{jw_{frame}\ell}$. The output from the imager is

$$\mathbf{y} = \mathbf{P}\mathbf{x} = \mathbf{P}\mathbf{A}\mathbf{f}(t) \tag{35}$$

where $\mathbf{P}$ is the matrix of sensor values. If the multiplication distorts the computation (i.e. from the differential transistor pairs), we can reformulate the result of second, third, and higher order harmonics by modifying $\mathbf{A}$ by $\mathbf{A}_1$, which takes these terms into account. Furthermore, we can invert this process to modify the starting matrix $\mathbf{A}$ to get a matrix $\mathbf{A}_1$, which gives the desired transform of interest. The correction will depend on the desired transform.

3.3.6 Bandwidth of MATIA pixel

Frequency response and resulting harmonic distortion of these pixel elements are important. These measurements show that this pixel element shows little change from dc to 100Hz, which is the limit of the present off-chip current measurements. This frequency response will be dependent upon the incoming light levels. A corner frequency of 30Hz is observed for light intensities four orders of magnitude lower than that obtained from average room light. From these measurements, bandwidths upto 100kHz are possible using on-chip measurement techniques. This will be sufficient for a 1024x1024 imager performing full matrix operations at 60Hz image rate.

Since we are modulating the input pixel currents, one should consider the highest modulation frequency that a particular pixel can support. We define the bandwidth as the

difference of the highest frequency (i.e, the fastest generated signal) minus the lowest frequency (i.e, the frame rate or block rate); typically assuming the bandwidth as related to the highest frequency is sufficient. This maximum frequency/bandwidth defines a trade-off between the resulting frame rate and the number of available pixel elements. We are looking at the frequency response for a differential signal, therefore, the source node of the differential pair is nearly fixed. Sensor capacitance and any capacitance in parallel with the phototransduction sensor have negligible effect on the frequency response.

For example, for a 1 million pixel imager (1Kx1K pixel array), we need 60KHz modulation for a 60Hz frame rate. If the current output lines use one-stage active feedback (as used in the adaptive photoreceptor [28]) to reduce capacitive effects, then we could approach these frequencies for 10pA of sensor current. A limit of 10pA significantly limits the range of input illumination, for lower currents either the image size must decrease or the frame rate must slow down accordingly.

We can reduce this minimum current level by using stronger active feedback or by changing the phototransduction method in the pixel cell. Stronger active feedback will improve the frequency response at a given current, and therefore reduce the minimum current that can be modulated. However, the stronger active feedback requires more gain, and therefore more power consumed and increased stability issues. One can change the phototransduction element to a vertical BJT to amplify the current, but this approach results in a more than proportional increase in the element noise, as well as decreases in pixel-circuit fill factor. Experimental measurements have qualitatively verified these results.

Often, early levels of image processing are based upon block transforms rather than full image transforms, and the bandwidth behaves similarly. For block processing, we often turn on a basis block when being used, and turn it off when not being used. The frequency response of turning on or turning off a block is fairly quick for both operations. Turning on the block, which means we are bringing up the resulting output voltage, looks like a source follower using nFETs on the upswing. That is, we are working on the fast transition region of this circuit. Turning off the block, which means we are pulling down the resulting output voltage, looks like we quickly drop the gate voltage below the source voltage, and therefore,

the current through the differential-pair FETs is very small.

3.4 Conclusion

In this chapter we have introduced the MATIA pixel. The pixel has a fill factor of 46% and can perform focal–plane multiplication. This pixel can be used either as a pixel for image readout or for performing various focal–plane processing. The pixel has a pitch of 45λ in a 0.5μ N-well CMOS process. It was also fabricated in 0.25μ N-well CMOS and has been characterized for proper operation.

Large arrays of this pixel has been fabricated for characterizing this pixel using 0.5μ N-well CMOS. We have characterized the pixel for dark currents, signal to noise, gain and offset mismatch across an array, linearity and harmonic distortion, and bandwidth. We have also discussed how some of these errors can be eliminated when this pixel is used as an imager.

CHAPTER IV

FOUR–QUADRANT CURRENT–MODE VECTOR–MATRIX MULTIPLIER

4.1 Overview of vector–matrix multiplier

Vector-matrix multiplication(VMM) is the fundamental operation in a lot of signal processing based computations. It is an important step in developing a variety of analog signal processing techniques such as 2-D block transforms for image processing systems [42], and FIR filtering [118]. The basic vector-matrix multiplication is defined as :

$$Y_j = \Sigma_i W_{ji} I_i \tag{36}$$

An analog implementation of such an operation can be compact, low power, and can eliminate data conversion in case of analog interfaces. On the other hand, a digital realization of this operation is both area and power intensive for a reasonably sized array, thus making it impractical for large VLSI systems [42]. Also, the computation can be done in parallel and faster in analog since the weights stored at each multiplier site saves the fetch time [6, 47].

Previous implementations have used some modification of EEPROM cells [68] or some variation of multiple-input floating-gates for analog storage [89]. The programming schemes used were slow and inaccurate. On the other hand, our adaptive programming technique allows for fast and accurate programming [76].

There have been various proposed implementations for the analog multiplication operation in voltage-mode. These implementations had limitations such as the use of MOS transistors in triode that are sensitive to drain-source variations [75], MOS transistors operating in saturation based on 'quarter-square algebraic identity' that used at least 12 transistors [103], dual-input floating-gate MOS that requires two capacitors per cell and have offsets in the final results that have to be corrected for off-chip [6]. Along with these, the maximum linearity available in voltage-mode implementations is limited up to power

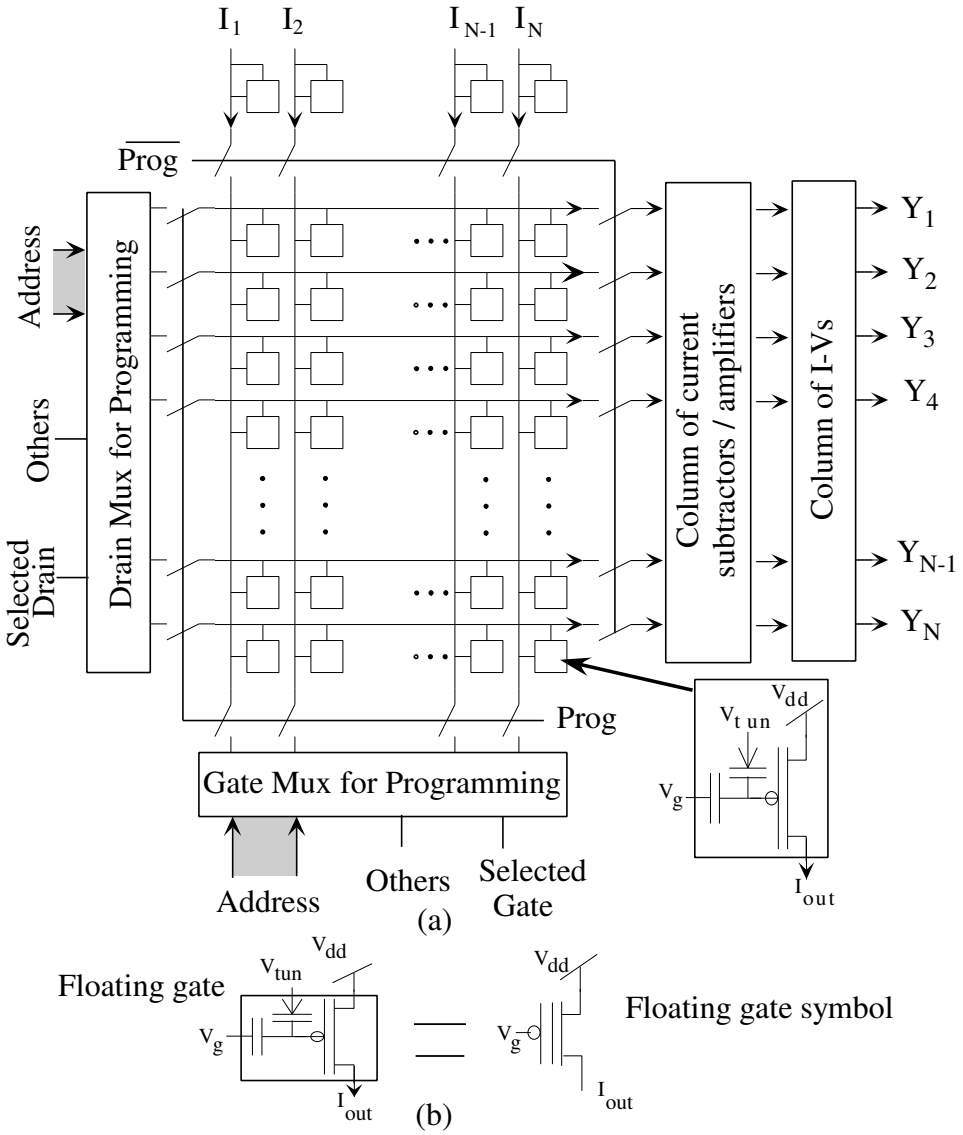


Figure 47: Block diagram of chip: (a) The chip consists of a 128x32 array of floating–gate vector matrix multiplier elements, peripheral digital control for isolation of floating–gate elements during programming, and current amplifiers; (b) Symbol used or a floating–gate (FG) device.

supply rails. All of these implementations operated at slow speeds and had high power consumption, which can be a limiting factor in some of the portable high-speed applications like video processing.

In order to achieve high power efficiency and low power operation, a sub-threshold implementation is ideal. A voltage-mode implementation operating in sub-threshold will have limited linearity due to the exponential I-V relationship of the transistor operating in saturation. This limitation in linearity can be alleviated to a certain extent by using

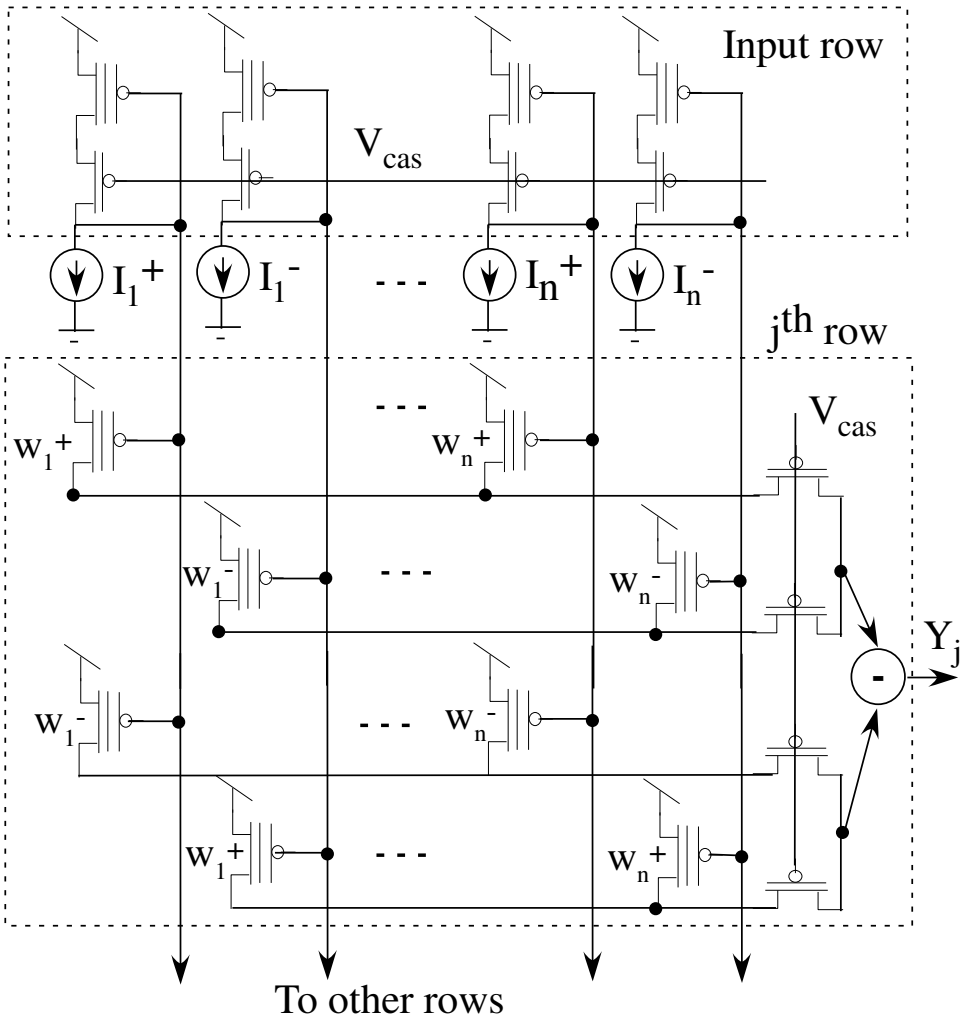


Figure 48: Schematic of four–quadrant current–mode multiplier: Circuit schematic showing the j^{th} row for a fully–differential current–mode vector–matrix multiplier.

methods like source degeneration, which degrades the frequency response for a particular power. A current-mode implementation can be used to overcome some of these limitations.

This chapter introduces this current–mode VMM. Section 4.2 introduces the architecture of this four quadrant multiplier. Section 4.3 discusses different issues like programming the VMM, frequency and speed of operation. We also present results for 8x8 block DCT on 128x128 images using this architecture. We conclude by summarizing the performance of this architecture.

4.2 Current–mode vector–matrix multiplier

Fig. 47 shows the block diagram of our programmable current-mode VMM architecture using floating-gate (FG) elements. The input vector values I_i are multiplied along each column by the stored weight W_{ji} and the results are summed along each horizontal row. The weights are stored as charge on a floating-gate transistor and the results Y_i are available in parallel along each row. Digital logic consisting of switches, decoders and multiplexors are used to isolate any individual floating-gate transistor for programming. Also, to aid in measurement, the output currents are amplified and then converted into a voltage-mode signal using linear I-V Converters.

Fig. 48 shows a detailed circuit schematic of the VMM system. The proposed multiplier makes use of a floating gate current mirror with the two floating-gates programmed to different charges. The difference in charge ($\Delta V_{charge,ji}$) represents the intrinsic stored weight with which the input signal gets multiplied. The weight is given by,

$$W_{ji} = e^{-\kappa(\Delta V_{charge,ji})/U_T} \tag{37}$$

where κ is defined as the variation of the surface potential with the gate voltage and U_T is the thermal voltage given by kT/q. In this fashion, the multiplier provides a non-volatile weight storage that is intrinsic to the structure and can be easily programmed to any desired value. In a floating-gate device, the output impedance is degraded primarily due to the drain voltage (V_d) variation coupling onto the floating-gate node through C_{gd} rather than Channel Length Modulation; cascoding helps in reducing the C_{gd}-coupling effect by making the drain of the floating-gate a low impedance node while maintaining a high impedance at the output. Also, the cascode transistors can be used as switches in the program mode to better isolate the elements and thus, serve a dual purpose.

Our VMM chip affords the flexibility of configuring the system as either a two-quadrant or a four-quadrant multiplier for both positive and negative weights. Different rows were programmed to different weights and all the weights in one particular row were programmed

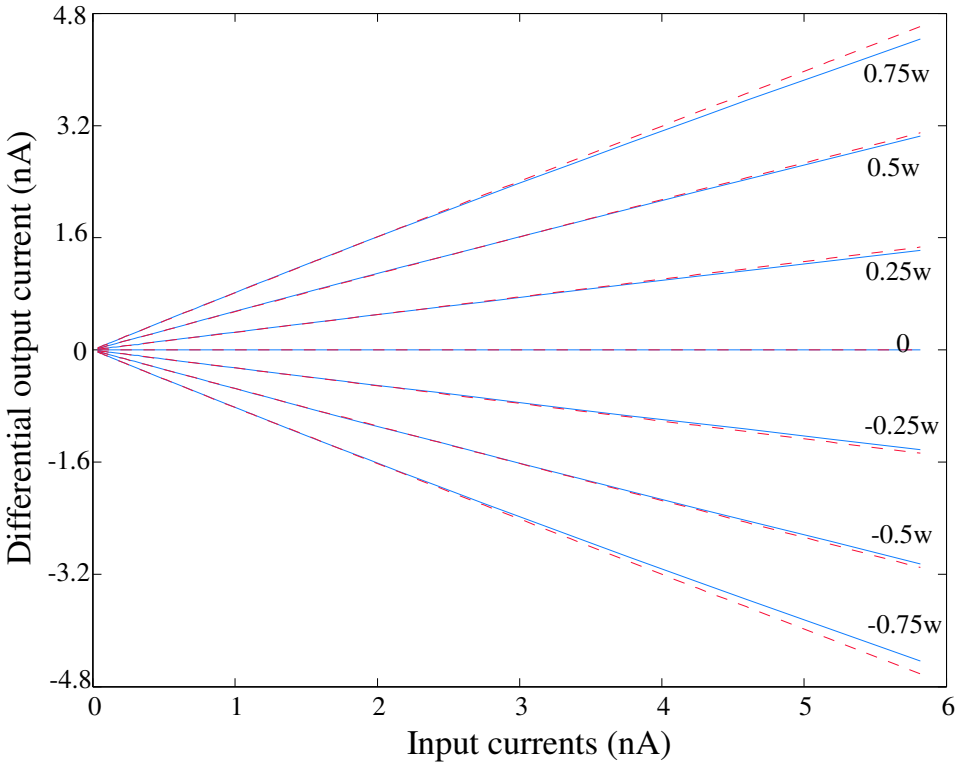


Figure 49: Measurements from two–quadrant current–mode multiplier: Plot of measured differential output current vs. input current on a linear scale, for the pseudo–differential configuration.

identical. Figure 49 and 50 demonstrate the functionality as a two-quadrant and four-quadrant multiplier respectively. Four-quadrant operation eliminates output DC offsets on-chip and helps improve linearity, which is evident from Figure 49 and 50 and the equation given below:

Pseudo–differential:

$$\begin{aligned} Y_j &= \Sigma[(w_{ji}^{+} - w_{ji}^{-})I_i^{+} + (w_{ji}^{+} - w_{ji}^{-})\Delta I_i^{+} \\ &\quad + \frac{(w_{ji}^{+2} - w_{ji}^{-2})\Delta I_i^{+}}{2} + \frac{(w_{ji}^{+3} - w_{ji}^{-3})\Delta I_i^{+}}{3}] \end{aligned} \tag{38}$$

Fully–differential:

$$\begin{aligned} Y_j &= \Sigma[(w_{ji}^{+} - w_{ji}^{-})(\Delta I_i^{+} - \Delta I_i^{-}) \\ &\quad + \frac{(w_{ji}^{+} - w_{ji}^{-})^3((\Delta I_i^{+})^3 - (\Delta I_i^{-})^3)}{3}] \end{aligned} \tag{39}$$

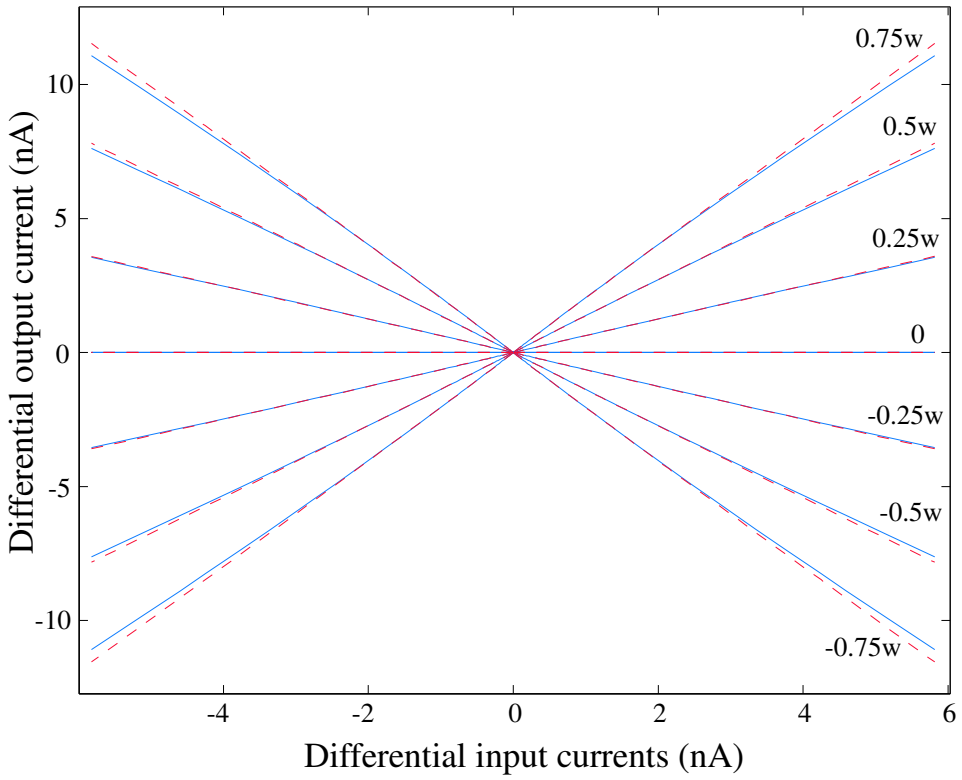


Figure 50: Measurements from four–quadrant current–mode multiplier: Measured differential output current vs. differential input current for the fully–differential configuration.

The linear range of the multiplier can be estimated from Figure 51 which shows the differential output current vs. the input current for various positive weights. Different weights show up as different intercepts in the log-scale. Linearity is clearly limited on the lower level of currents, which is not evident from the previous plots. The linearity is measured to be greater than two decades, beyond which the multiplier deviates from the ideal linear curve with an error that is higher than 2.5%. This linearity limitation is partly due to the difference in κ between identical transistors programmed to different currents and the variation of κ with the gate voltage. This effect can be alleviated by programming the elements relatively close to each other. Fig. 51 also emphasizes the point that a current-mode implementation gives decades of linearity in signal swing that is especially hard to obtain in voltage-mode circuits without consuming more power. For instance, in [6], a linear range of 1V - 4V is obtained at the expense of 0.39mW of power dissipation. Figure 52 shows the linear range for a voltage–mode implementation. In the current–mode implementation,

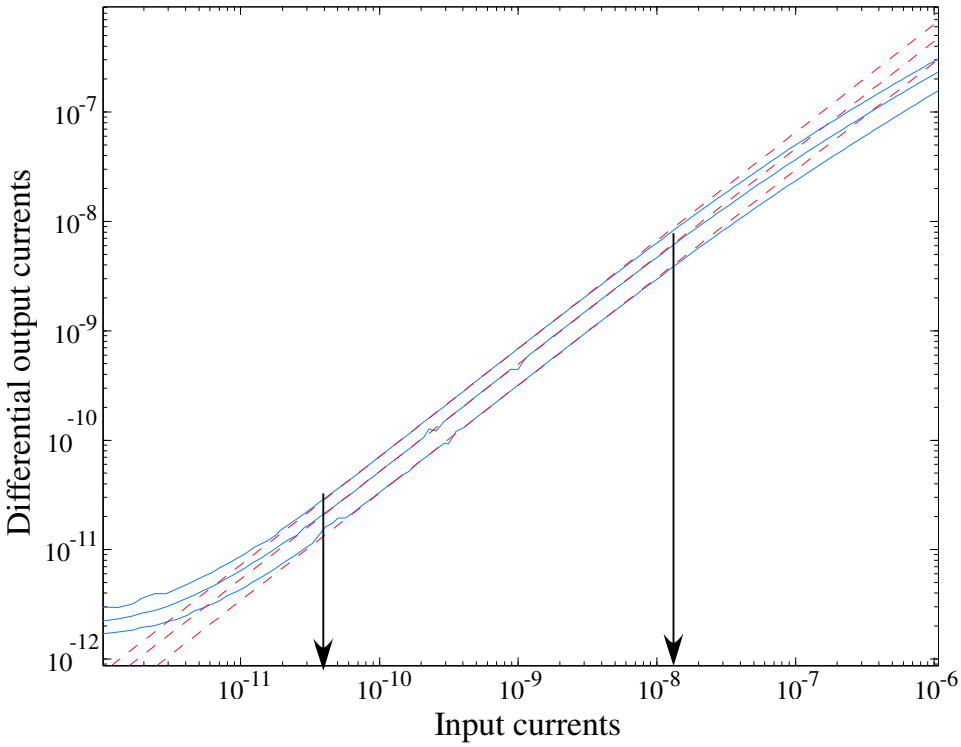


Figure 51: Linearity of four–quadrant current–mode multiplier: Plot showing the limits of linearity for the current–mode configuration for the pseudo–differential configuration.

the DC level of input current determines both the speed and the power dissipation and can be programmed to any desired value.

4.3 Experimental results, applications and discussions

4.3.1 Programming VMM

FG devices, as shown in Fig. 47, are a good option to implement programmable systems. One of critical aspects in the design of programmable VMM system is programming accuracy. Previous implementation using FGs used programming scheme similar to that used for EEPROMs based on electron tunneling [6]. This method required special oxide and at least a dual gate implementation adding extra fabricating steps. It also required an extra switch per element to select the cell to be programmed, along with the decoders, and thus increasing area/cell [6]. The method was based on giving small pulses of constant drain-to-source voltages (V_{DS}) which affected the accuracy of programmed weights.

Our programming scheme is based on using both hot-electron injection and electron

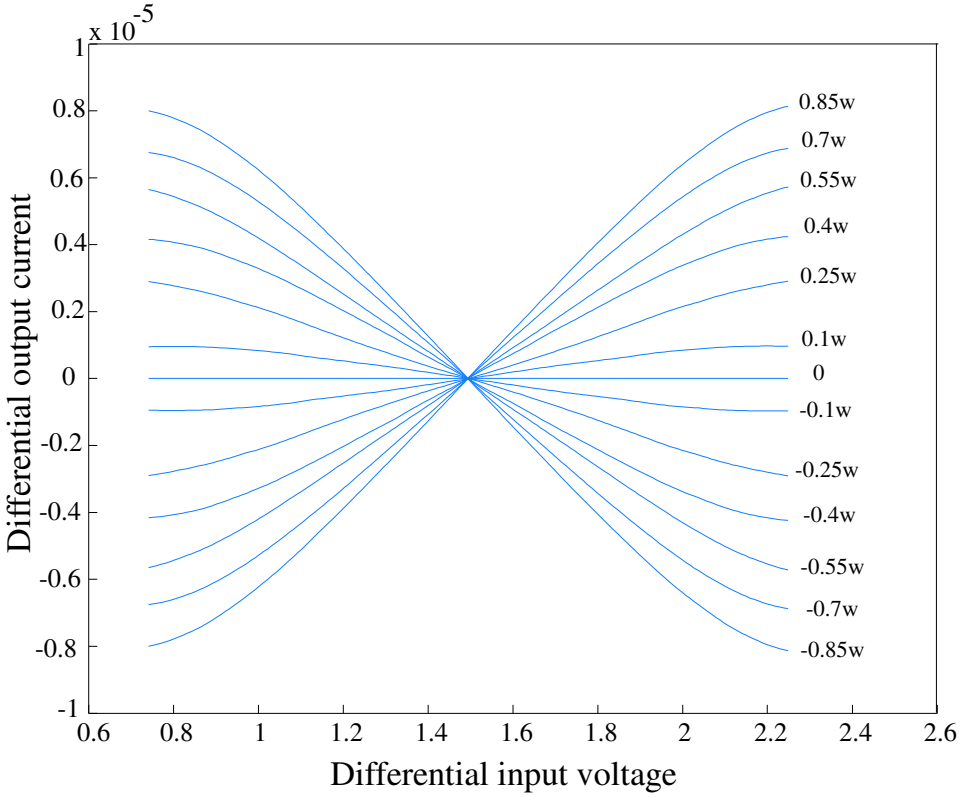


Figure 52: Voltage mode VMM: Measured differential current output vs. differential input voltage for a voltage–mode configuration.

tunneling. We do not need any special oxide or extra gate to program these devices enabling easy integration in typical CMOS processes. Our adaptive programming method enables us to perform accurate and fast programming [76]. The algorithm computes the V_{DS} steps based on the device current and target current. This value is adjusted automatically as the device current approaches the target current. Number of steps required to hit a target are on average 10-15 pulses. Fig. 53(a) shows sine-wave coefficients programmed on 128 elements of a single row. Percentage error plot between the programmed value and the target current,as shown in Fig. 53(b), gives a 0.2% of worse case deviation.

4.3.2 Frequency and speed measurements

A custom PCB was fabricated to perform speed measurements for low input currents. An on-chip current amplifier with variable gain along with an off-chip I–V converter was used for taking measurements especially with lower currents. Figure 54(a) shows the measured

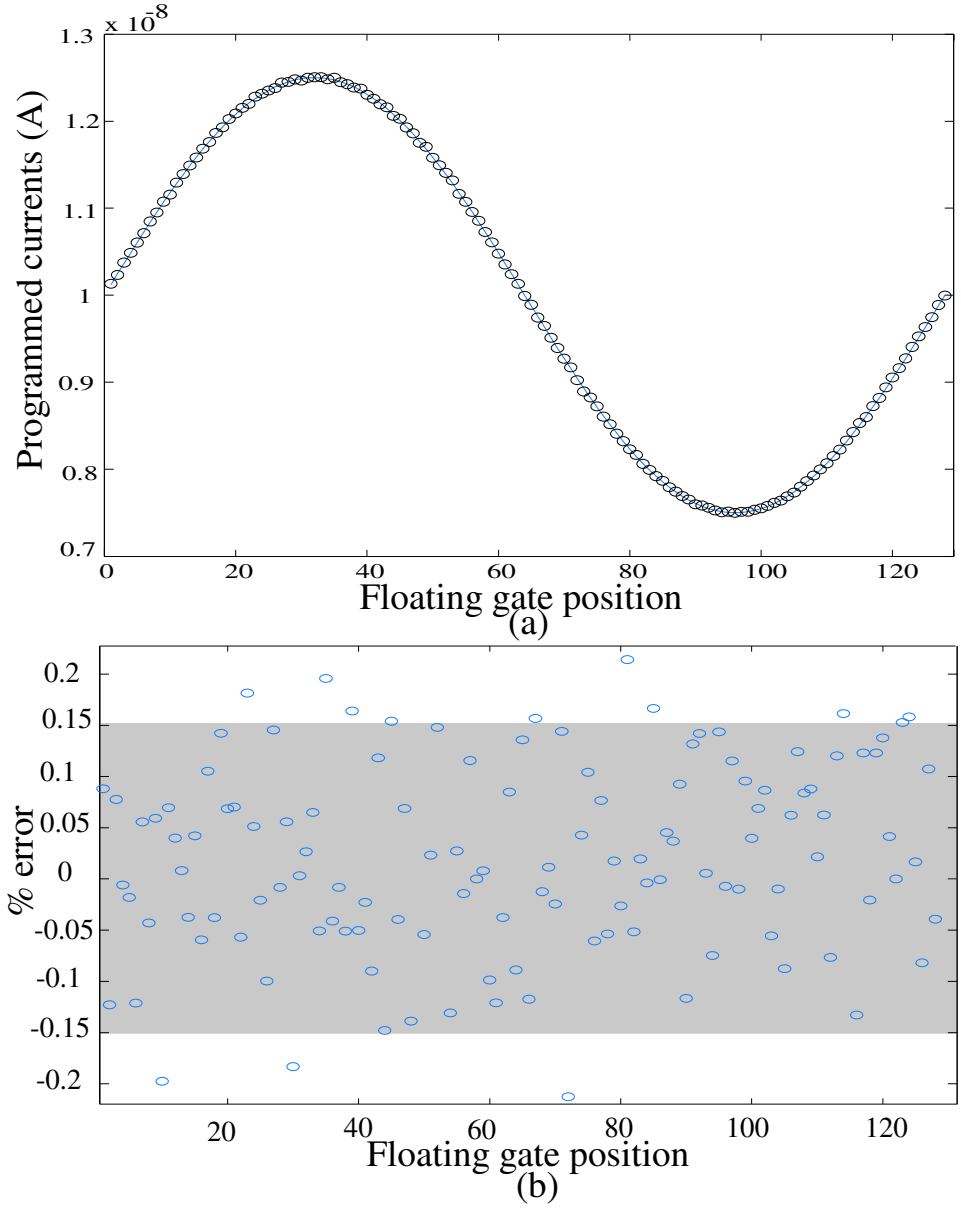
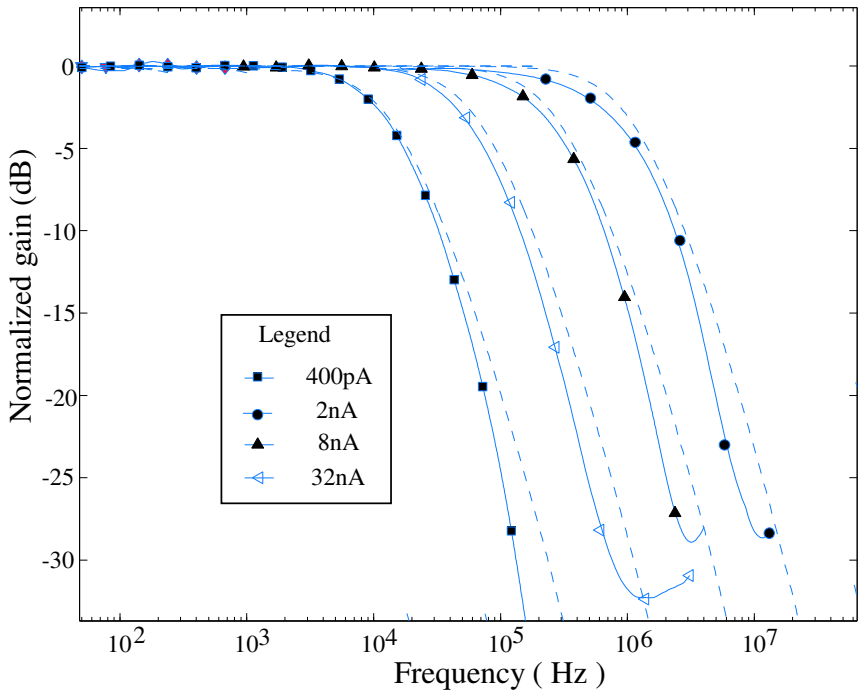
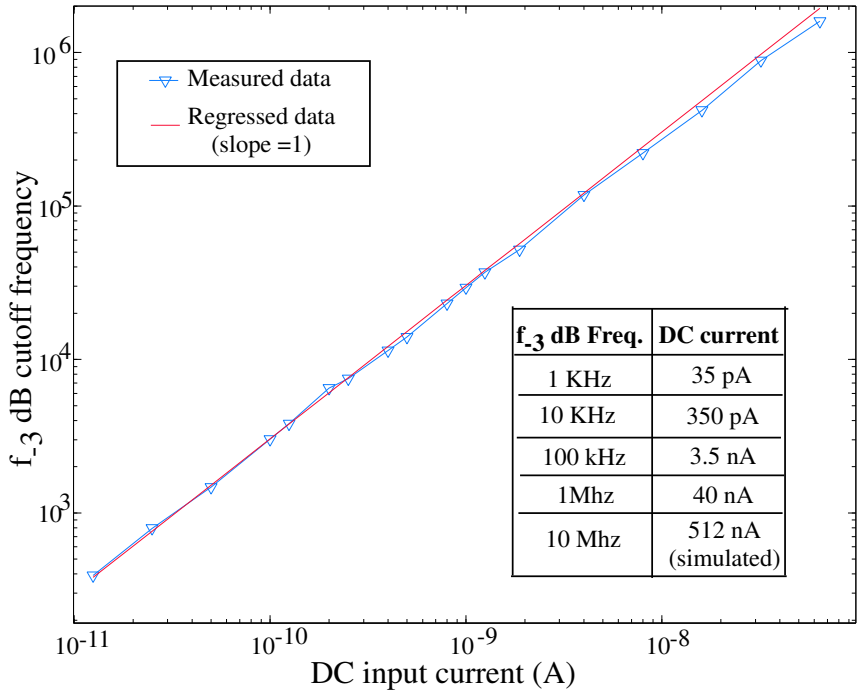


Figure 53: Programmed values: (a) A sine wave with 5 nA p–p and a DC of 10 nA was programmed onto 128 floating–gate elements; (b) Percentage error per element is shown.

and simulated frequency response for different DC input currents. The measured corner frequencies (f_{-3dB}) match closely to the simulated results. The plot shows that the VMM would easily operate up to 10MHz if it was not limited by the frequency response of the I–V converter (Bandwidth = 5MHz) at the output. Figure 54(b) shows a plot of measured corner frequencies with the input DC bias current on a log-log scale. The data points follow a straight line with a slope of 1 as expected in sub-threshold. The deviation for



(a)



f_{-3} dB Freq.	DC current
1 KHz	35 pA
10 KHz	350 pA
100 kHz	3.5 nA
1Mhz	40 nA
10 Mhz	512 nA (simulated)

(b)

Figure 54: Frequency response: (a) Plot of frequency response of current mode multipliers. The solid lines represent measured data while dashed lines represent simulation results; (b) Variation of f_{-3dB} cut–off frequency vs. DC input current (per FG device) is plotted. For subthreshold currents a linear relationship is observed, as expected. The table shows the measured DC input current (per FG device) required for various f_{-3dB} cut–off frequency.

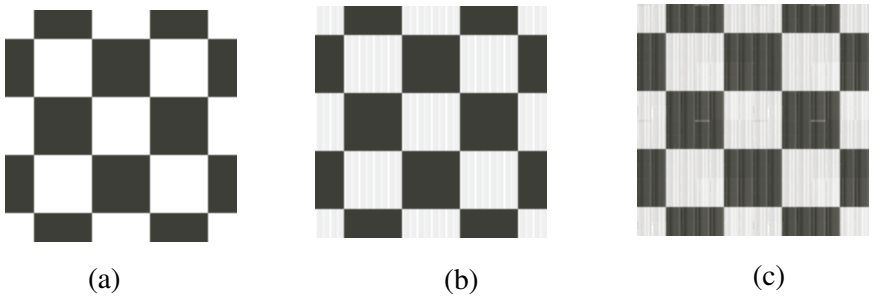


Figure 55: 8x8 block DCT of a 128x128 image: (a) Original input image; (b) Image after inverse DCT, when block matrix transformation was performed off–chip, using the measured weight matrix from the VMM chip. (c) Output of the VMM chip (after inverse DCT) for 8x8 block transform that was performed on–chip.

higher current levels is due to the transistor moving from sub-threshold regime to the above threshold region. The bias currents required for a bandwidth of 1MHz and 10MHz are 40nA (measured) and 512nA (simulated), respectively for each FG device.The VMM chip required 531nW/MHz (from Fig. 54(b)) for each differential cell clearly demonstrating the speed vs. power tradeoff. The DC bias current however can be set solely on the basis of speed requirements as the Signal-to-Noise Ratio (SNR) is independent of the input DC bias level. The SNR however is directly proportional to the Gate-Source Capacitance (C_{gs}) and can be increased at the expense of chip area. A PCB was fabricated to perform the measurements.

4.3.3 Performing DCT using VMM

The proposed VMM architecture can be used to perform real–time block matrix transforms of input images in a row–parallel manner as proposed in [7]. We have demonstrated DCT using the VMM as it is used for various applications. Fig. 55(a) shows the 128x128 image that was placed as an input to the chip. For a fair comparison of the multiplier, we have shown the output (after inverse transformation) when the programmed weights were measured and the block DCT (8x8) was performed off–chip in Fig. 55(b). Fig. 55(c) shows the image obtained, after inverse transformation, when the block transform (8x8) was performed on–chip. It can be observed that the results for part (b) and (c) are similar. The distortion observed in both the images are due to the programming accuracy limitations

Table 2: Summary of performance for proposed VMM

Parameter	**Proposed VMM**	**VMM in [6]**
Technology	0.5μm N-Well CMOS	1.5μm single poly CMOS/EEPROM
Power Supply	3.3V	5V
FG Dim.(W/L)	18λ / 4λ	N/A
Array size	128×32	16×16
Chip area	0.83mm^2	1mm^2
Programming % error	$< \pm 0.2\%$	<10mV
BW/power per cell	531 nW/MHz	N/A
Linearity	$>$ 2 decades	3V
Power per cell	7.2 μW @10MHz	0.39mW @60KHz
Programming scheme	Hot–electron injection and Tunneling	Electron Tunneling
Programming Time per W_{ji}	1mS	100mS

Figure 56: Block diagram of chip: The VMM chip consists of a 128x32 array of floating–gate elements, current amplifiers, and peripheral digital control for isolation of floating–gate elements during programming.

(0.2% error).

Table 2 summarizes the performance of our VMM along with that of [6]. As can be observed, the proposed architecture is both power and area efficient. Fig. 56 shows the micrograph of the VMM chip that was fabricated in a 0.5μm N-well CMOS process.

4.4 Conclusion

A programmable fully differential current mode VMM architecture has been presented in this chapter. Fig. 56 shows the micrograph of the VMM chip that was fabricated in 0.5μm N-well CMOS. Table 2 summarizes the chip performance. We have demonstrated block

matrix transforms using this architecture. The architecture is suitable for low power applications and has bandwidth-to-frequency ratio of 531nW/MHz per differential multiplier. For a bandwidth of less than 10MHz, this architecture is capable of performing 1 million MAC/0.9μW as compared to a commercially available DSP (TMS32005x series), which gives 1 million MAC/0.25mW. The VMM chip can be used for applications like audio and video processing.

CHAPTER V

MATRIX TRANSFORM IMAGER ARCHITECTURE (MATIA)

5.1 Digital image compression

Digital image processing involves storing images in short–time memories for convenient access, processing them on a pixel by pixel basis, and achieving the processed data or transmitting the results to another location. However, the demand for handling more and more data continues to outpace advances in microelectronics technology and upgrades in telecommunications infrastructure. Digital image compression represents an immediate and practical approach to help address storage limitations and transmission channel bottlenecks. If images can be stored in compressed form then the amount of stored data on a given storage device can be increased by orders of magnitude, depending on the methods used. For transmission applications if images can be compressed a similar improvement can be realized.

Images can be compressed because they contain redundancies. By removing these redundancies the size of the image can be reduced, i.e. the number of bits needed to represent the image can be reduced. From a mathematical viewpoint this amounts to transforming a 2-D pixel array into a statistically uncorrelated data set. In digital image compression, three basic data redundancies can be identified and exploited: *coding* redundancy, *interpixel* redundancy, and *psychovisual* redundancy [48]. Data compression can be achieved when one or more of these redundancies are removed. The transform is applied prior to storage or transmission of the image. Later the compressed image is decompressed to reconstruct the image or an approximation of it.

The compression can be either lossless or lossy. The advantage of lossless compression is that the accuracy of representation is preserved accurately. Hence, this type of compression

is useful in applications like image archiving (e.g. legal or medical records) as images or data can be stored and retrieved without any loss of data. The disadvantage is that the amount of compression is very limited. To achieve higher compression, factors of 10–to–1, some amount of redundancy has to be tolerated. This type of compression is called lossy compression. Even with lossy compression, in most applications the distortion can be made so small that the image quality is more than acceptable after reconstruction. Lossy image compression is useful in applications like broadcast television, video conferencing, and facsimile transmission, in which a certain amount of error is acceptable trade-off for increased compression performance [48].

Traditionally these compression have been performed in the digital domain. These compression methods use block transforms, which essentially mean repeated matrix multiplications between the compression kernel and the image sub–block. Matrix multiplication involves element–by–element multiplication and then sum of products along a column. Performing these operations in analog domain can reduce power consumption and also space required, as compared to these being performed in the digital domain. In a previous chapter we had presented a pixel that can be used for matrix multiplication. The element–by–element multiplication is performed at the pixel itself, while the addition takes place by Kirchhoff's current law (KCL) because of the way the pixels are connected together. In this chapter we will focus on a MAtrix Transform Image Architecture (MATIA) that uses the above mentioned pixel and performs image compression (baseline JPEG).

This chapter describes the MATIA system and also the experimental results obtained from it. Section 5.2 describes separable transforms and the various kernels that can be stored on the MATIA. Section 5.3 describes the system architecture including the on–chip bias generation and kernel storage, four–quadrant current–mode multiplier, on–chip I–V for image readout, peripheral circuits used for this system and also floor planning and isolation issues. Section 5.4 illustrates how an on–chip flash can be used for faster readout. Section 5.5 and Section 5.6 deals with the PCB that was fabricated for testing purposes and the various on–chip image transforms that can be performed using MATIA. Low–power JPEG/motion JPEG compression implementations using MATIA are described in

5.7. The chapter concludes by presenting a summary of performance of this system–on–a–chip camera.

5.2 Separable transforms

Separable transforms are those in which operations can be applied separately to the rows and columns. Consider the following general transform equation:

$$y[k_1, k_2] = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x[n_1, n_2] \phi_{n_1,n_2}[k_1, k_2] \tag{40}$$

The transform is separable if the kernel $\phi_{n_1,n_2}[k_1, k_2]$ can be expressed in the separable form $\phi_{n_1}[k_1]\phi_{n_2}[k_2]$. If so then Eq. 40 can be be written as:

$$y[k_1, k_2] = \sum_{n_2=0}^{N-1} g[k_1, n_2] \phi_{n_2}[k_2] \tag{41}$$

where:

$$g[k_1, n_2] = \sum_{n_1=0}^{N-1} x[n_1, n_2] \phi_{n_1}[k_1] \tag{42}$$

Thus the new image (after inner summation), $g[k_1, n_2]$ is a 1-D transform in the variable n_1. N of these N–point transforms are performed, one for each value of n_2. The outer summation is a series of 1-D transforms in the variable n_2, one for each value of k_1. Hence, separable implementation involves N 1-D transforms in n_1, and another N transforms in n_2, resulting in a total of 2N 1-D transform evaluations.

MATIA can compute arbitrary separable 2-D linear operations. These operations are expressed as two matrix multiplications on the image,

$$Y = A^T P B \tag{43}$$

where P is the image array of pixels, Y is the computed output image array, and A and B are the transform matrices [48]. The values of A and B are stored in analog floating-gate arrays for on–chip processing.

In image processing, the most common linear operations consist of FIR filtering and real transforms such as the discrete cosine transform (DCT) or wavelet transforms. Examples of the left–side matrices, $\mathbf{A}^T$, for these operations are shown in Figure 57. For the FIR

$$\begin{bmatrix} h_{0,0} & h_{0,1} & h_{0,2} & h_{0,3} & h_{0,4} & h_{0,5} & h_{0,6} & h_{0,7} \\ h_{1,0} & h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & h_{1,5} & h_{1,6} & h_{1,7} \\ h_{2,0} & h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} & h_{2,5} & h_{2,6} & h_{2,7} \\ h_{3,0} & h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} & h_{3,6} & h_{3,7} \\ h_{4,0} & h_{4,1} & h_{4,2} & h_{4,3} & h_{4,4} & h_{4,5} & h_{4,6} & h_{4,7} \\ h_{5,0} & h_{5,1} & h_{5,2} & h_{5,3} & h_{5,4} & h_{5,5} & h_{5,6} & h_{5,7} \\ h_{6,0} & h_{6,1} & h_{6,2} & h_{6,3} & h_{6,4} & h_{6,5} & h_{6,6} & h_{6,7} \\ h_{7,0} & h_{7,1} & h_{7,2} & h_{7,3} & h_{7,4} & h_{7,5} & h_{7,6} & h_{7,7} \end{bmatrix}$$

(a)

$$\begin{bmatrix} h_{0,0} & h_{0,1} & h_{0,2} & h_{0,3} & 0 & 0 & 0 & 0 \\ h_{1,0} & h_{1,1} & h_{1,2} & h_{1,3} & 0 & 0 & 0 & 0 \\ h_{2,0} & h_{2,1} & h_{2,2} & h_{2,3} & 0 & 0 & 0 & 0 \\ h_{3,0} & h_{3,1} & h_{3,2} & h_{3,3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_{0,0} & h_{0,1} & h_{0,2} & h_{0,3} \\ 0 & 0 & 0 & 0 & h_{1,0} & h_{1,1} & h_{1,2} & h_{1,3} \\ 0 & 0 & 0 & 0 & h_{2,0} & h_{2,1} & h_{2,2} & h_{2,3} \\ 0 & 0 & 0 & 0 & h_{3,0} & h_{3,1} & h_{3,2} & h_{3,3} \end{bmatrix}$$

(b)

$$\begin{bmatrix} h'_0 & h'_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_{-1} & h_0 & h_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & h_{-1} & h_0 & h_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_{-1} & h_0 & h_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_{-1} & h_0 & h_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_{-1} & h_0 & h_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_{-1} & h_0 & h_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & h'_{-1} & h'_0 \end{bmatrix}$$

(c)

$$\begin{bmatrix} h_{0,0} & h_{0,1} & h_{0,2} & h_{0,3} & h_{0,4} & h_{0,5} & h_{0,6} & h_{0,7} \\ h_{1,0} & h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & h_{1,5} & h_{1,6} & h_{1,7} \\ h_{2,0} & h_{2,1} & h_{2,2} & h_{2,3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_{2,0} & h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,0} & h_{3,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_{3,0} & h_{3,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_{3,0} & h_{3,1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{3,0} & h_{3,1} \end{bmatrix}$$

(d)

Figure 57: Image transform matrix examples: The transform imager can perform many types of operations of the type $\mathbf{Y} = \mathbf{A}^T\mathbf{PB}$ where $\mathbf{A}^T$ operates on the columns of the image $\mathbf{P}$ and $\mathbf{B}$ operates on the rows. Examples of $\mathbf{A}^T$ are shown here for different types of operations. (a) A transform of the entire image where $h_{i,j}$ represent the windowed transform basis elements. (b) Block transform of the type more likely to be used in image compression. (c) FIR filter applied to the image, (d) Wavelet transform of the image, note a block wavelet transform could also be applied.

filter (Figure 57(c)) note that the corner coefficients are denoted with $'$'s because they are often normalized to account for the shorter length of the filter at that point; or they may be changed to accomplish filtering of a symmetrically extended image with $h'_0 = h_0$ and $h'_1 = 2h_1$, etc.

The range of operations possible within the architecture, which is expressed in Eq. 43, is significant. For example, it is possible to use differentiating FIR filters to do better edge detection or lapped orthogonal transforms for image compression without blocking artifacts. Smoothing filters combined with a decimation scheme could provide simple data reduction. Arbitrary transforms can be considered, because computational complexity and efficient algorithms are not a concern. Additionally, cascaded operations can be performed by collapsing the matrices describing the multiple operations:

$$\begin{aligned} \mathbf{Y} &= \mathbf{C}^T \left[\mathbf{A}^T \mathbf{P} \mathbf{B}\right] \mathbf{D} \\ &= \hat{\mathbf{A}}^T \mathbf{P} \hat{\mathbf{B}} \end{aligned} \tag{44}$$

where $\hat{\mathbf{A}} = \mathbf{AC}$ and $\hat{\mathbf{B}} = \mathbf{BD}$.

Note that even though arbitrary matrices can be used without considering traditional computational complexity, the connectivity complexity should be considered. For example, a full image transform requires the instantiation and routing of the full transform matrices while a block transform can be implemented using only enough elements and interconnects for the non-zero transform matrix elements.

The basis matrix has the following structure:

$$A = \begin{pmatrix} v[0,0] & v[0,1] & \cdots & v[0,N-1] \\ v[1,0] & v[1,1] & \cdots & v[1,N-1] \\ \vdots & \vdots & \ddots & \vdots \\ v[N-1,0] & v[N-1,1] & \cdots & v[N-1,N-1] \end{pmatrix} \tag{45}$$

where,

$v[n,k]$=$C_k \cos[\frac{(2n+1)kn}{2N}]$, for DCT-II;

$v[n,k]=\sqrt{\frac{2}{N+1}}\sin[\frac{\pi(k+1)(n+1)}{N+1}]$, for DST.

The Hadamard transform can be written in the matrix format as:

$$A_{2^n} = A_{2^{n-1}} \otimes A_2 = A_2 \otimes A_{2^{n-1}} \tag{46}$$

where,

$$A_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{47}$$

Similarly Haar and Slant transforms can also be written in the matrix format for different sizes [48, 102]. To read an image one needs to program the kernel A to an unitary matrix so that,

$v[n,k]= 1$, for $n = k$; else $v[n,k]=0$.

The MATIA chips can also be used for image enhancement using spatial filters. Various sized smoothing filters (lowpass or averaging filters) can be programmed onto the chip. This can be used to reduce "sharp" transitions in gray levels. Since random noise typically consists of sharp transitions in gray levels, the most obvious application of smoothing filters is noise reduction. Unfortunately this method also blurs the edges (sharp transitions in gray levels), and this is an undesirable side effect of these filters. Examples of these filters are as follows:

$$A = \frac{1}{4}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{48}$$

$$A = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{49}$$

Similarly sharpening spatial filters can be implemented too. These are essentially spatial differentiation filters. The simplest isotropic derivative operator is the Laplacian. The filter

mask can be as follows:

$$A = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{50}$$

A high–boost filtering technique can be implemented using the following kernel:

$$A = \begin{pmatrix} 0 & -1 & 0 \\ -1 & b+4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{51}$$

where b $\geq$ 1. When b = 1, high boost filter becomes the "standard" Laplacian filter. As b increases, the contribution of the sharpening process becomes less and less important. Eventually, if b is large enough, the high–boost image will approximately equal to the original image multiplied by a constant.

5.3 Architecture description

This section describes the system architecture for our transform imager. The transform imager architecture is both modular and programmable making it ideal for image data–flow computations. This approach allows for retina and higher–level bio-inspired computation in a programmable architecture that still possesses high fill–factor pixels characteristic of APS imagers. Figure 58 shows the block diagram of MATIA.

This imaging architecture is made possible largely by advancements in analog floating–gate circuit technology [63]. These circuits have the added advantage that they can be built in standard CMOS or double-poly CMOS processes. In this approach, the floating–gate circuits store and reproduce arbitrary analog waveforms for image transforms, allow for correction factors to account for device mismatch, as well as perform matrix-vector computations.

The basis functions for the various block matrix transforms are stored on-chip using floating gates. Each floating gate can be individually programmed to a store one coefficient. Peripheral digital control allows for individual floating gates to be selected and programmed to any arbitrary current for any gate voltage. The mechanisms involved in this programming

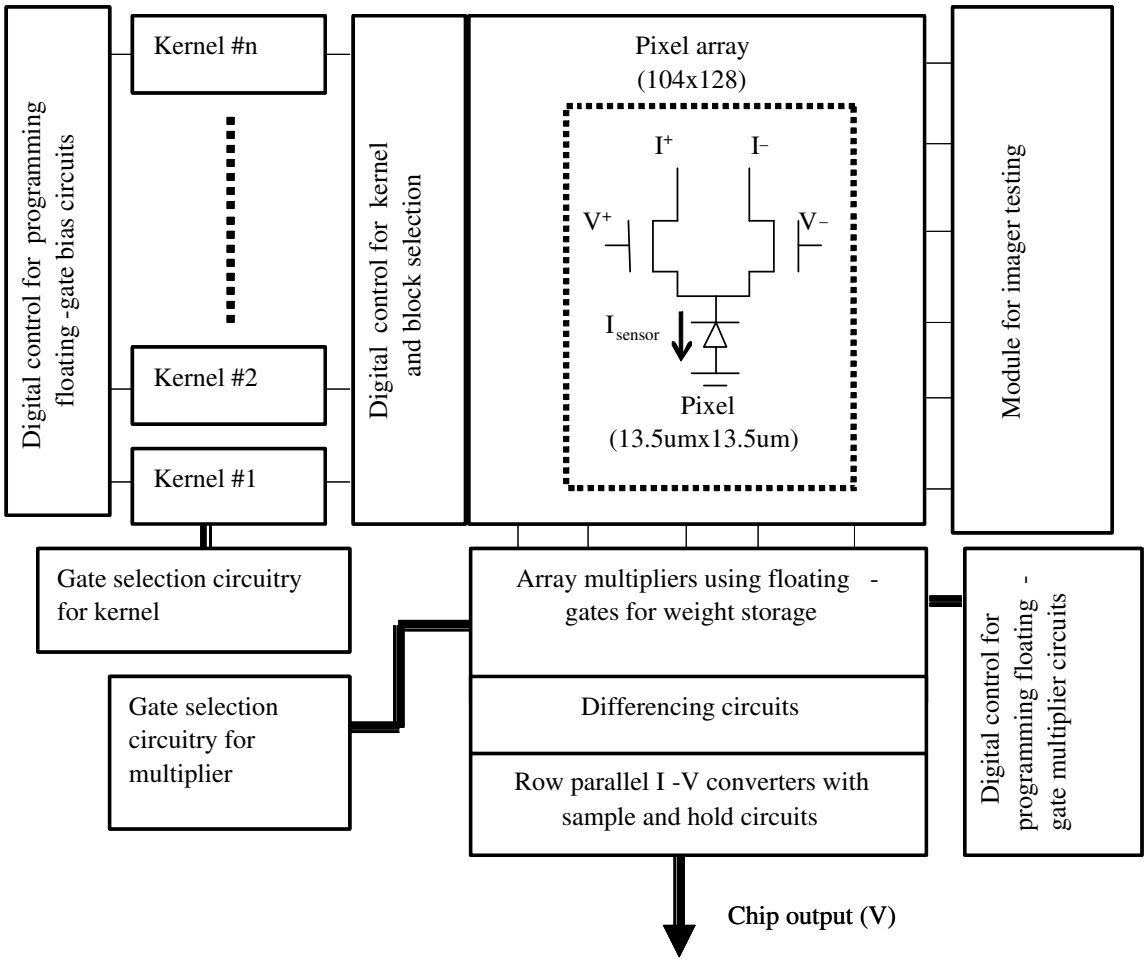


Figure 58: Top view of MATIA: This programmable architecture performs arbitrary separable block matrix image transforms. Each pixel processor multiplies the matrix coefficients with the measured image sensor output. The basis functions are stored on–chip using floating–gate circuits.

phase are electron tunnelling and hot electron injection. In the transform phase, the drains of the floating gates are connected to current–to–voltage (I–V) converters which convert the respective currents to the bias voltages required for the transform. The imager array is designed such that multiplications can occur at the pixel level. The corresponding output is then placed in a parallel fashion to the vector matrix multiplier (VMM) for the second matrix multiplication. The VMM consists of an array of floating gates for further multiplication. For the second multiplication floating gates are used instead of normal pFETs to reduce area and also perform arbitrary multiplications. These floating gates can be programmed to multiply incoming currents by any arbitrary factor depending on the transform being

performed.

The architecture was designed such that different kernels can be programmed onto the chip at once. The kernels can be applied individually to the imager using digital control. These kernels can also be used in a feedback loop (with the computer in the loop for now) to fine tune a kernel for better performance. They can also be used in the "ping–pong" method where one kernel is being used for image transformation, while the other one is being fine tuned. The roles can be switched when the fine tuning process is complete. This method can be used to reduce noise and enhance performance of the imager on the fly, depending on the kernel and the image.

The output currents are transduced to voltages by means of integrating type I–V converters. These have been deigned for column parallel operation. The resultant voltage is sampled and held, after the integration has been performed for a fixed amount of time, using open loop sample and hold circuits. The opamp for the I-V is a wide swing two stage opamp using high swing cascodes for better PSRR, and noise performance. The biases required for the opamp are generated on chip using power supply independent bias circuits. The values are then read sequentially using on–chip decoders. These can then be digitized by either using on–chip methods or using on–board ADCs.

The peripheral circuits include:

- Clock distribution circuits.
- Decoders for isolation and accurate programming of floating gates
- Decoders for image readout and other controls
- On–chip drivers

The architecture's scalability makes it feasible to compute large scale digital camera resolution images. Furthermore, MATIA allows for data reduction that is compatible with machine vision and biological modelling, as the computation is performed in the focal plane. This architecture can be directly extended to a number of applications like depth computation from stereo, motion estimation by optical flow methods, spatial and temporal filtering

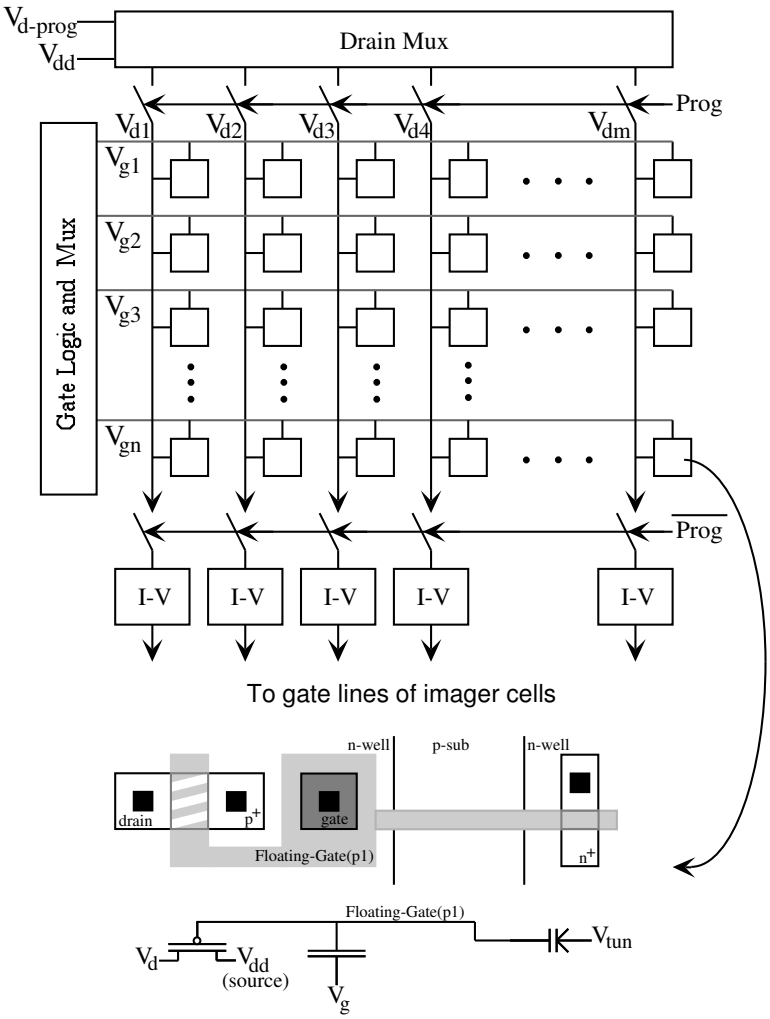


Figure 59: Top-level view of our basis generation circuitry: In operation (transform) mode we have an array of stored values that are output in sequence. Low-pass filtering on the output results in a continuous-time analog result. In programming mode we can easily reconfigure this circuitry on the outside edges for programming.

[53]. The next few sections give detailed description of the various sub–blocks of the MATIA system.

5.3.1 On–chip bias generation

Floating-gate circuit elements are used to store and generate arbitrary basis functions needed for the matrix-vector multiplication on the imager [62, 56, 60]. Figure 59 shows the top-level view of our basis generation circuitry. In programming mode, we can easily reconfigure this circuitry on the outside edges for programming. This approach is compatible

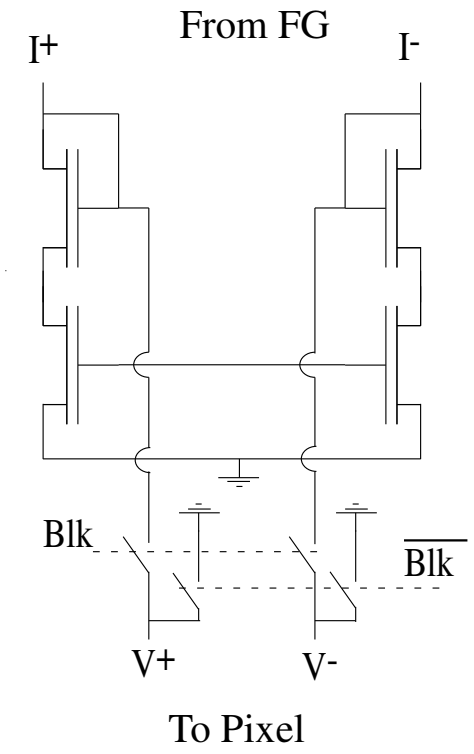


Figure 60: Schematic of current–to–voltage converter: This circuit is used to convert FG generated current to the corresponding bias voltage to be transmitted to the pixel. The bias voltages (V^+ and V^-) are turned off if the block is not chosen.

with our standard programming structure and algorithm. Each floating gate can be isolated for programming using the peripheral digital control circuits. The array of floating gates are initially tunnelled so that they have almost negligible currents for the operating gate voltage. They are then programmed, using hot electron injection, to the various currents needed for the bias generation. The floating gates can be used to store arbitrary waveforms. The floating gates are programmed so that they supply different currents for the same bias gate voltage. More details of floating gate programming can be found in Chapter 2.

In operation (transform) mode these currents are converted to the corresponding bias voltages (V^+ and V^-), which are required for different matrix transforms, before they are transmitted to the pixel array. We use voltage outputs because we can eliminate the effect of offsets from our images. Current outputs form a current mirror, and therefore differential pair offsets become gain errors. We also show the effect of varying the floating-gate bias voltage, resulting in significant changes of current to the current to voltage (I–V) converter,

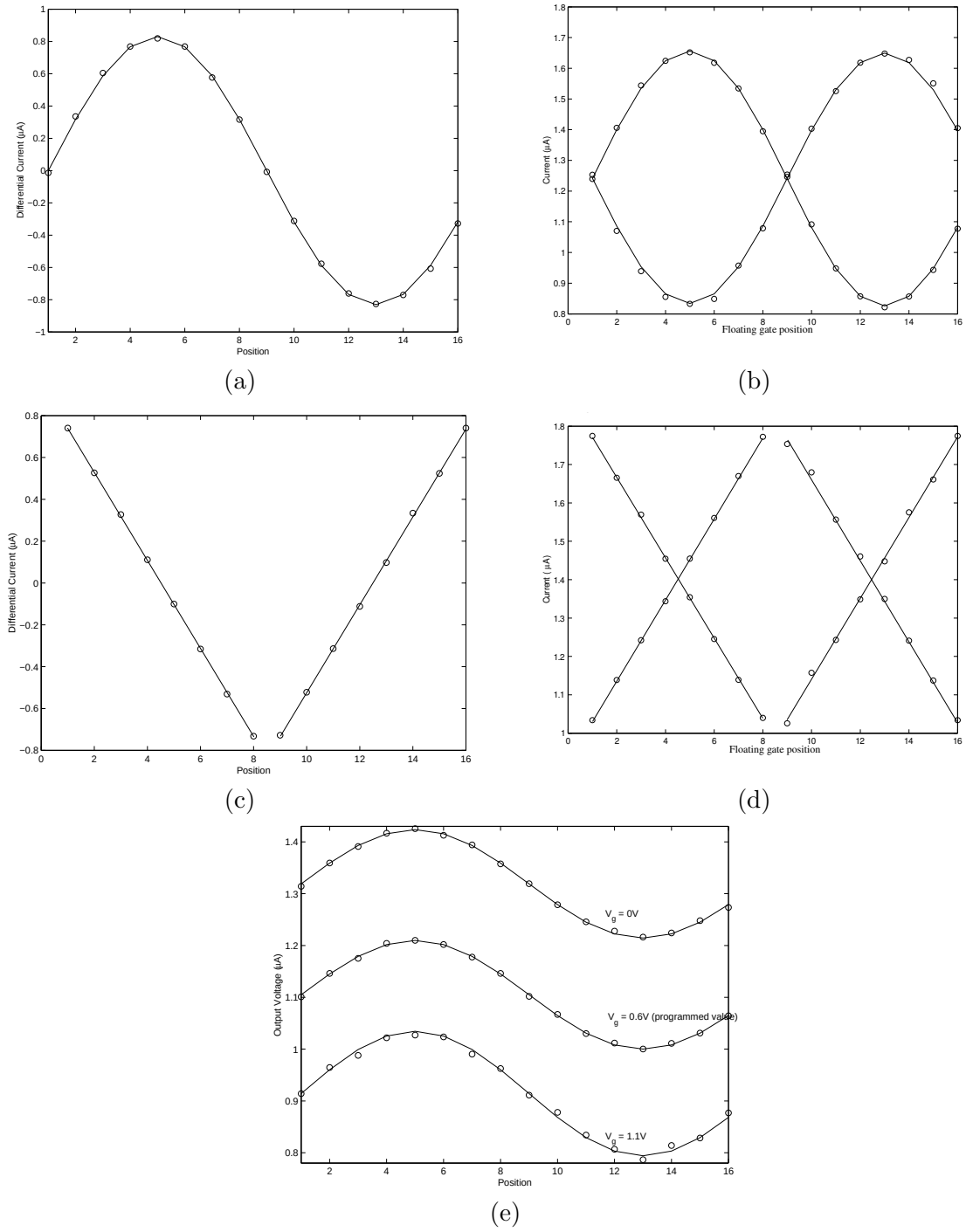


Figure 61: Bias generator outputs: (a) Sine wave (current), (b) Differential sinusoidal current waveforms, (c) Triangle wave (current), (d) Differential triangular current waveforms, (e) Programmed sinusoidal voltage outputs for different floating gate bias voltages.

resulting in a change in the bias voltage level. Figure 60 shows the schematic of the I–V converter used. These I–Vs are active only when the chip is in the transform mode. In the

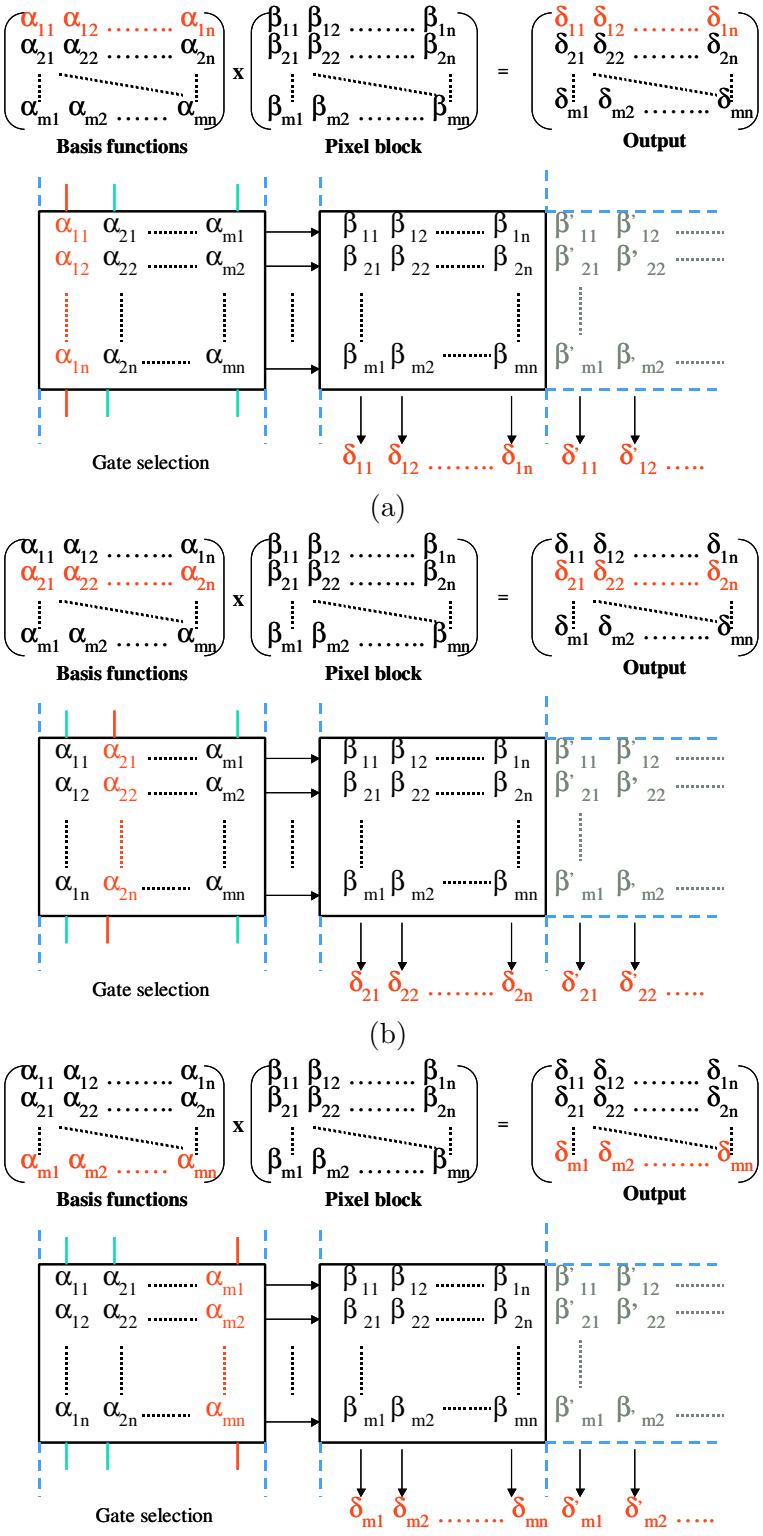


Figure 62: Coefficient storage and multiplication: Shows how the coefficients are stored, the sequence of steps for performing matrix multiplication, and the row–parallel operation of the MATIA system

program mode these are turned off because there is no path for the current, from the FGs, to flow. The outputs of the I–V are then placed on to the chosen pixel block by using digital control (block_sel). If a particular block is chosen then the outputs from the corresponding I–Vs are placed onto the V^+ and V^- busses for that particular block. If a block is not chosen it is turned off by connecting the V^+ and V^- busses to ground as shown in Fig. 60.

Figure 61 shows the programmed outputs for a single row of floating gates. Since the pixel structure is differential two voltages are required per pixel. The single ended and differential sinusoidal and triangular waveform are shown as examples of kernel storage. Figure 61(e) shows how after programming the DC value of a sinusoid can be changed just by changing the gate voltage with minimal distortion. As expected, since the FG is a PMOS, the DC increases with decreasing gate voltage, and vise versa.

Figure 62 illustrates how a basis function is actually stored for block matrix multiplication. For multiplying two matrices a row of the first matrix is selected. This is then multiplied on an element–by–element basis with all the columns of the second matrix. A sum–of–product of these multiplications give the corresponding row of the output matrix. Figure 62 also illustrates the sequence of digital control required for a block matrix transform. The transpose of the kernel (first matrix) is stored as shown in the figures. A corresponding row of the kernel is chosen by using gate selection (red when chosen, green otherwise). When a row is chosen the corresponding kernel coefficients are placed onto the V^+ and V^- busses. These are transmitted to all pixels of the chosen image block. Figure 62(a) shows how the first row of the output matrix is obtained in a column parallel fashion by just choosing the first row of the kernel matrix. Similarly the other rows of the kernel are chosen and the corresponding rows of the transformed image are readout.

5.3.2 Four–Quadrant current–mode multiplier

We use the floating-gate circuit elements to compute analog multiplications of a signal vector with a stored programmable matrix. We can perform vector matrix computations using current mode differential VMM. Using the output image stream this system will compute a transposed matrix transform.

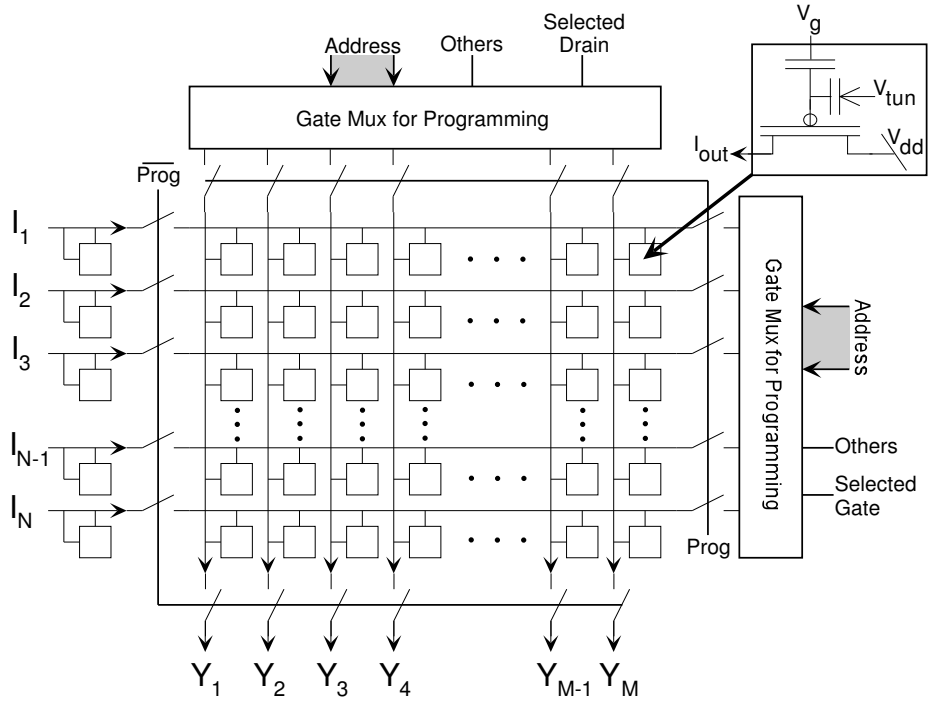


Figure 63: On–chip current mode multiplier: This programmable four quadrant multiplier operates at 3.3V supply, consumes 531nW/MHz and is linear for over two decades of current range. It consists of programmable floating–gate elements and hence is very area efficient.

Voltage implementations are possible, but they suffer from limited linearity, slow speeds, and high power consumption. A current–mode subthreshold implementation can achieve low power operation, high power efficiency and high linearity. Figure 63 shows the schematic of the differential current-mode VMM which was used in this system. It also shows the digital logic used while programming for isolation of individual FG transistors and accurate programming of the weights. This block is capable of over 2 decades of linearity (for a worst case error of 2.5%) and has a bandwidth of 1MHz for bias current of 40nA for each FG element. The floating-gate current mirror elements are cascoded for reducing channel length modulation effects, and increasing noise performance for the frequency of interest. It operates with V_{DD} of 3.3V, and the total area of this multiplier is $0.83mm^2$. It is capable of performing 1 million multiply–accumulate (MAC) operations per 0.27μW.

Figure 64 shows the schematic and the layout of the input stage of the current mirrors. This structure is connected as a diode–connection when the chip is in transform mode. When in program mode the the cascodes (pcas and ncas) is turned off and the drain pin

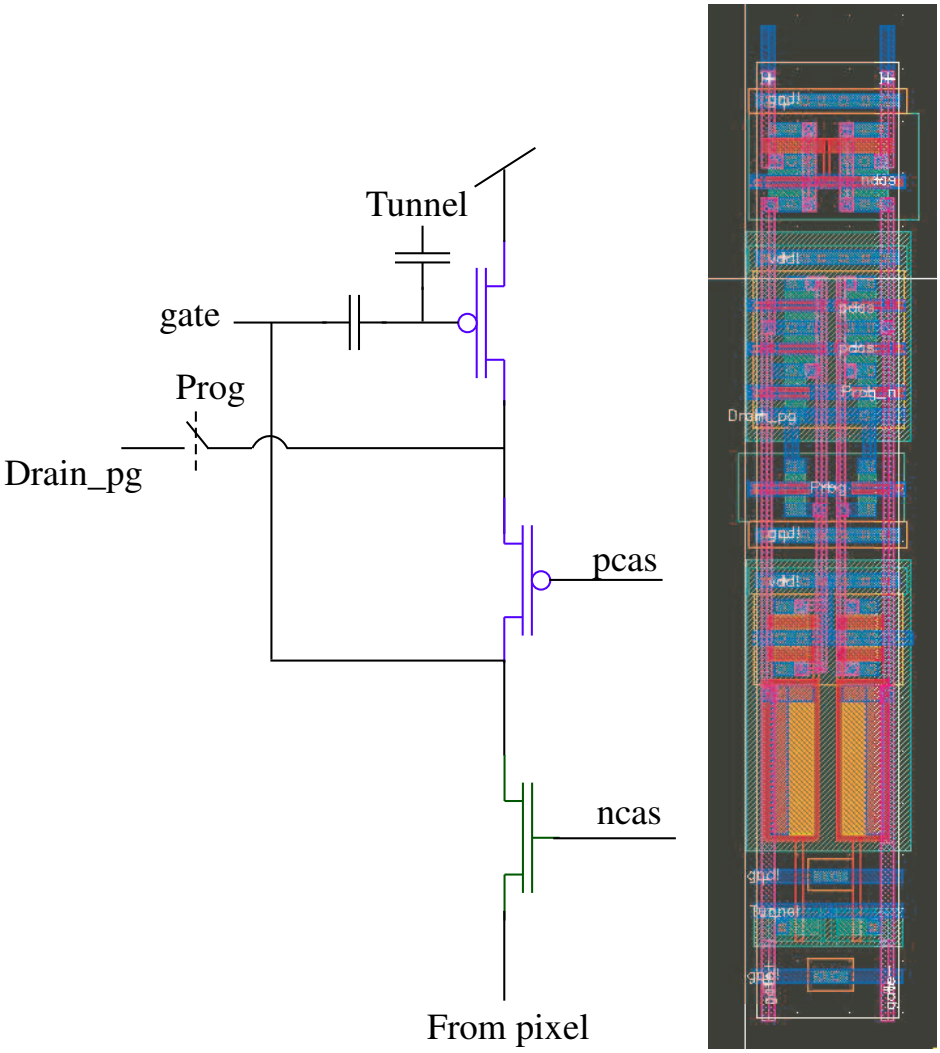


Figure 64: Input stage for current–mode VMM: This shows the schematic (a) and the layout (b) of the input stage for the current–mode VMM. It consists of a FG transistor, cascode transistors and switched that are used during programming.

of the FG is connected to Drain_pg. This pin can then be pulsed for injecting the FG or measuring the corresponding current through the FG.

Figures 65 and 66 show the schematic and the layout, respectively, of the output FG transistors of the VMM. In this case the cascode transistors are fully turned on so that the same drain pin can be used for programming the respective FG as well as for reading currents.

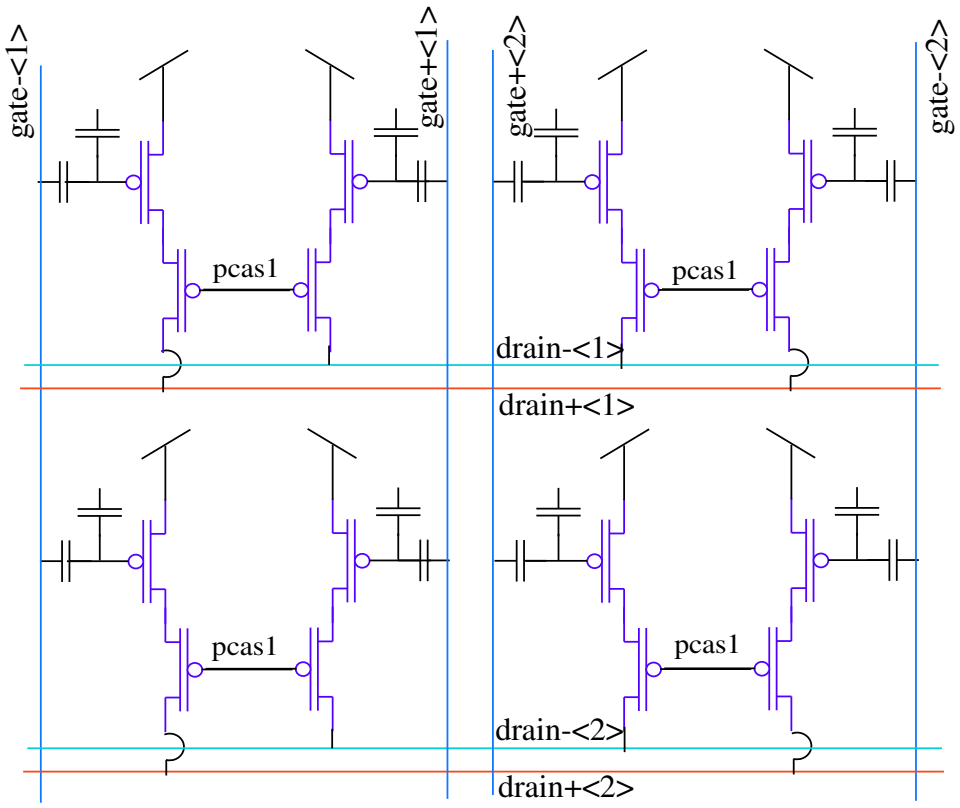


Figure 65: Output stage for current–mode VMM: The schematic of the output stage of the VMM is shown here. It also shows how the transistors are connected. It uses cascodes which are turned sully on during the program mode, while used as normal cascode transistors during operation mode.

5.3.3 On–chip single slope ADC

Figure 67(a) shows the integrating type I–V used for row parallel current readout. We use an open loop sample and hold, which is buffered out for readout. A wide swing two stage OTA is used as the integrating opamp. A wide range cascode structure is used for higher output swing and higher gain. The cascode biases are generated on–chip using temperature independent bias generators. The integrating capacitors were chosen such that they match across the array while enabling quick readout of low currents. The currents from the array are integrated at the same time (for a whole column) and their respective voltages are then readout sequentially during the hold phase. This enables parallel readout and reduces readout time. Each row of floating gates has a I–V converter for current readout. Figure 68 shows the layout of such a I–V converter. It has a guard ring around it for isolation. The output waveforms are shown in Fig. 69. The I–Vs are first reset and then allowed to

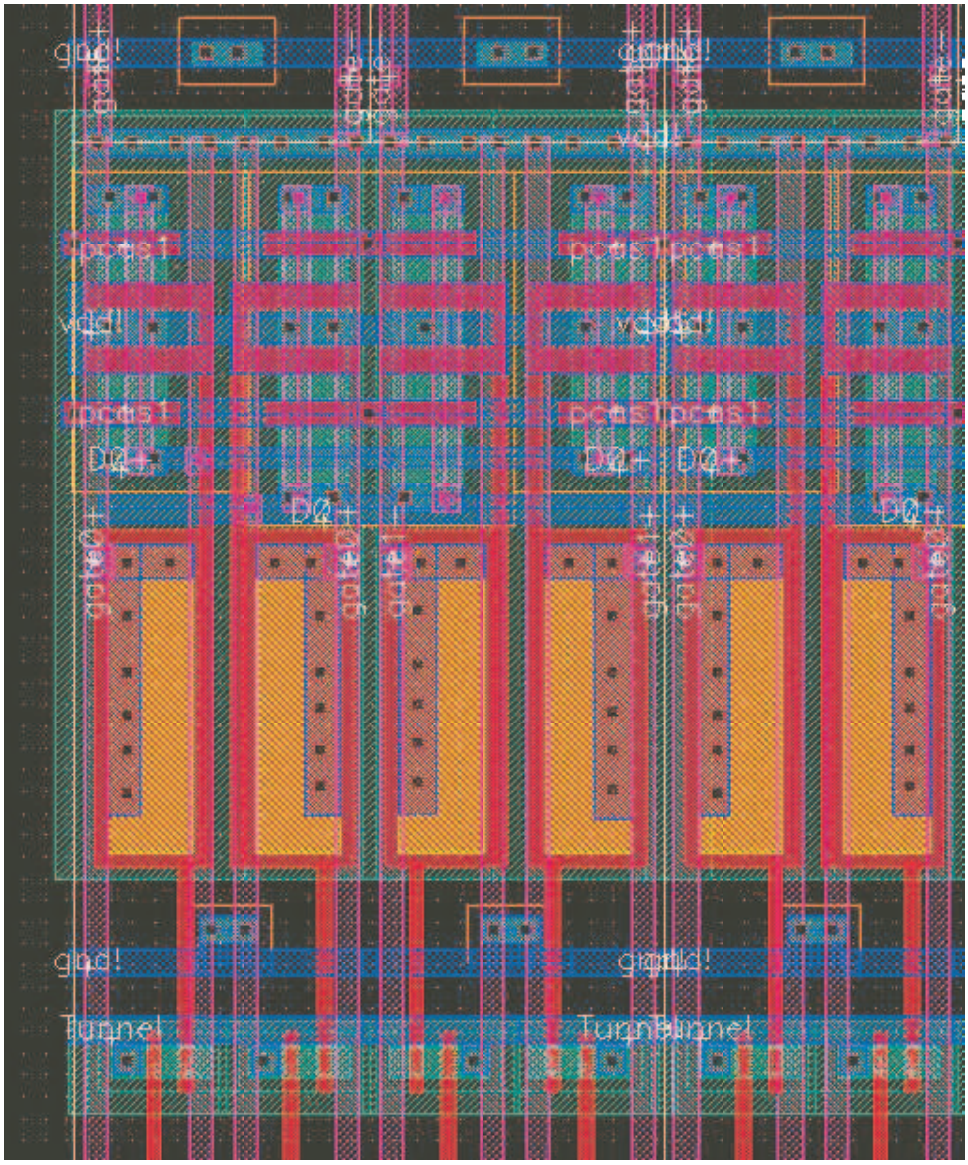


Figure 66: Layout of the output stage of the VMM: The layout of a part of the output stage of the VMM is shown here. It shows how each output stage was laid out such that they can be tiled in one direction.

integrate. At the end of the integration cycle the final value is sampled. The sample signal was designed to be active low as shown in Fig. 69.

5.3.4 Peripheral circuits

I have designed, simulated and tested on–chip clock buffers. These are used to *clean up* a digital signal coming into the chip from an FPGA through a bond pad. These were used as the inputs to various decoders and multiplexors used in the MATIA system. Figure 70

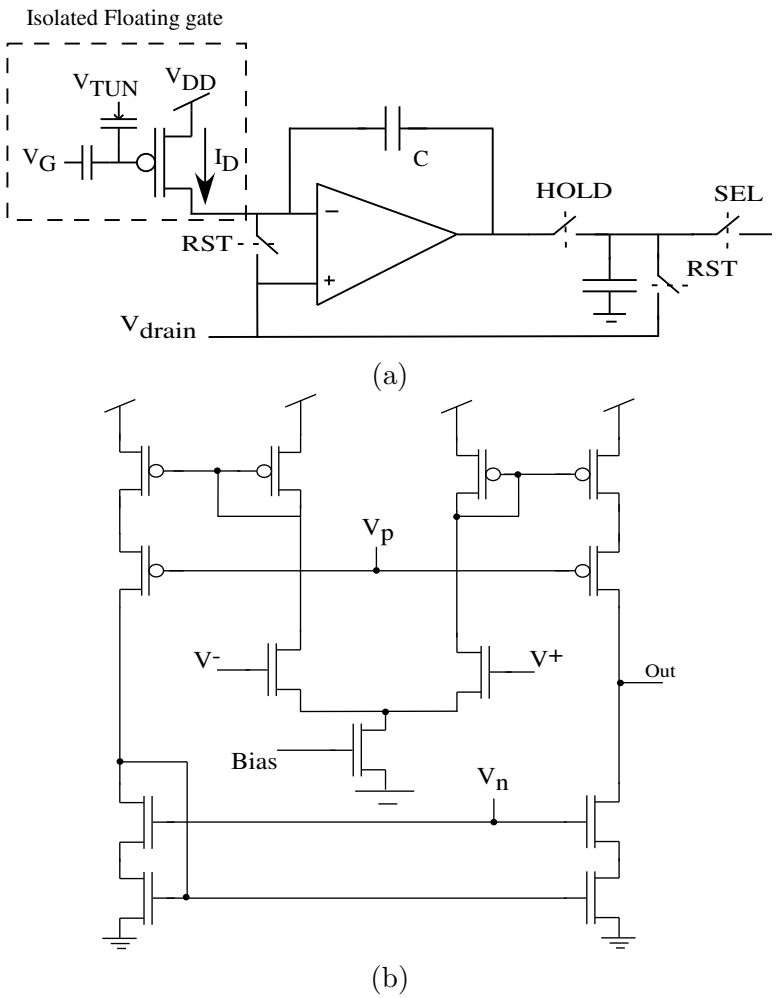


Figure 67: Circuit elements: (a) Shows the schematic for the integrator type I-V that was used for row parallel current readout. The schematic of a pFET floating-gate transistor is also presented. (b) Shows the schematic of the wide swing OTA that was used for the integrator.

shows the schematic and layout of the on–chip clock buffers. Since these digital signals would be driving large on–chip loads a multi–stage buffer is used. The ratio for minimum delay associated with a buffer is $e = 2.7182$ [96]. Upscaling by this ratio implies a large number of stages for driving a given load, hence I have chosen a ratio of three. This circuit also minimizes the propagation delay between clk and clk_n. To reduce ground bounce and isolation I have surrounded each clock buffer with guard rings. I have also designed digital buffers for driving off–chip capacitance (upto 100pF) with a speed of 40MHz. The layout was done using interdigitated fingers for big transistors. This was a four stage buffer, as shown in Fig. 71. I have used these circuits extensively for all the chips that I have designed

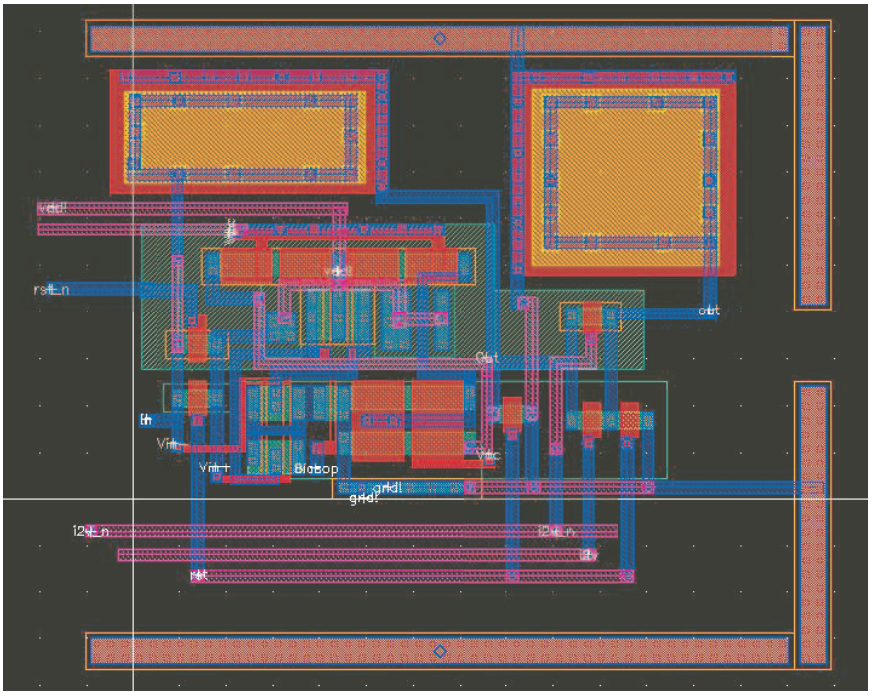


Figure 68: Layout of the integrating I–V: The layout of the integrating type single slope I-V along with an open–loop sample and hold circuit.

and fabricated.

NO-RAce logic targets the implementation of fast and pipelined datapaths using dynamic logic. A NORA datapath typically consists of a chain of alternating ϕ and $\overline{\phi}$. NORA design style can be simplified, so that single clock is sufficient to correctly operate dynamic sequential CMOS circuit [119]. The resulting design methodology is called *true single phase clock logic* (TSPC) because it allows for the implementation of dynamic sequential circuits with a single clock phase. Figure 72 shows the schematic and layout of a D flip–flop that was designed to fit a pitch of 45λ in one direction. The layout was done such that these can be tiled next to each other in pitch. This circuit has two clock connections hence the clock load is reduced, compared to other implementation needing more clock connections, especially for circuits employing many registers such as large shift registers.

When clk is high, the latch is in the transparent *evaluate mode* and corresponds to four cascaded inverters; hence it is noninverting. On the other hand, when clk is low, both the inverters are disabled, and the latch is in the *hold mode*. In this case only the pull–up

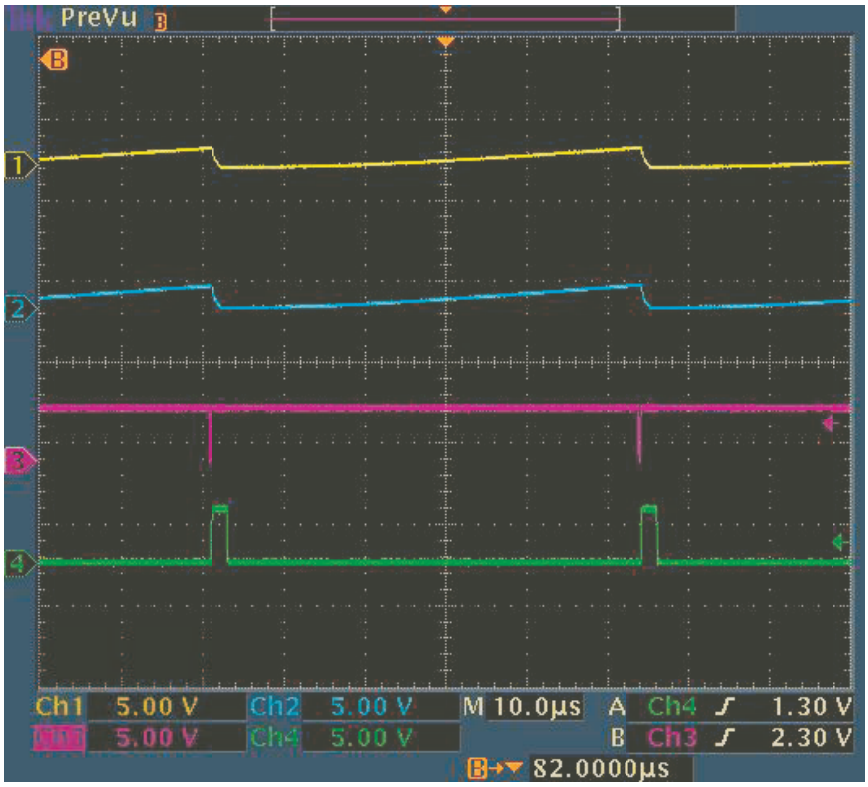
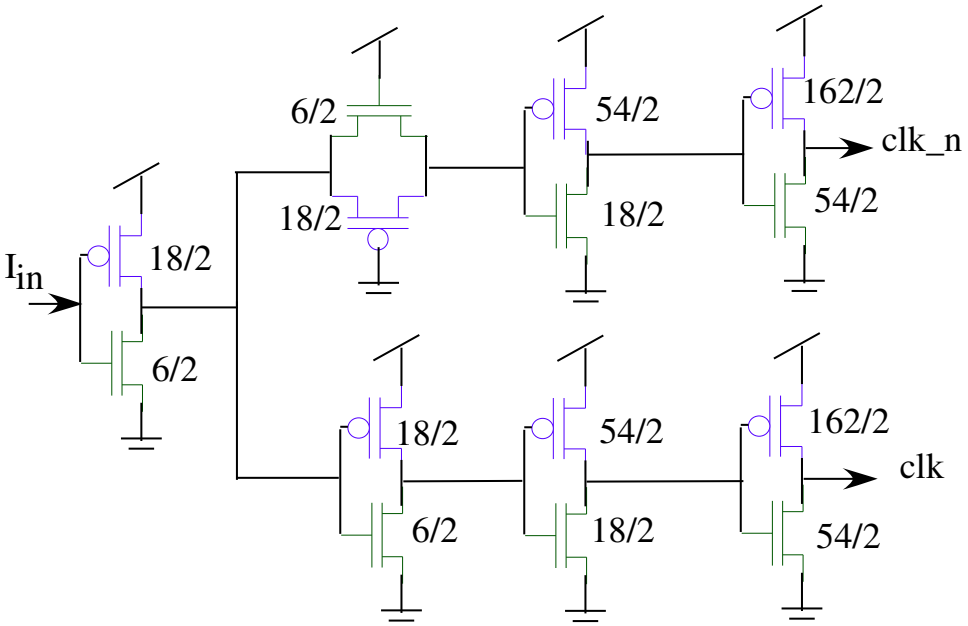


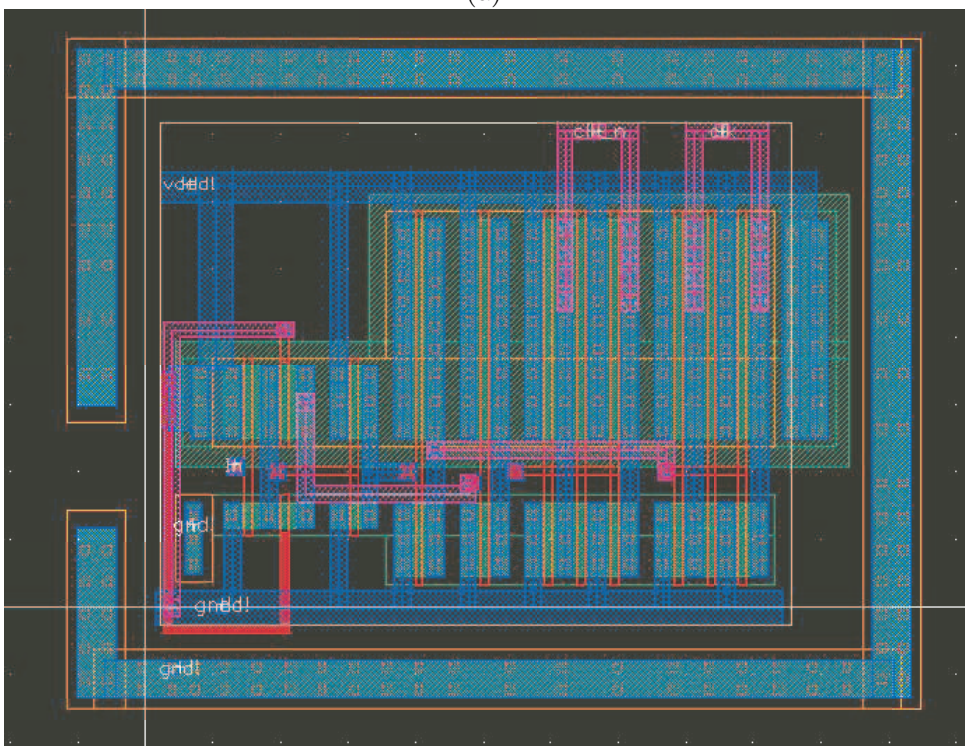
Figure 69: Operation of I–V: The output of two channels of I–V (channels 1 and 2) are shown above along with the sample (channels 3) and reset (channels 4) waveforms.

transistor are active. No signal can propagate from the input of the latch to the output in this operation mode. Hence, races are eliminated. During this operation (not reset mode) phi is connected to clk. During the reset mode the reset signal is connected to clk. The circuit is reset by forcing the input nodes of the second inverter to ground. The circuit was designed, simulated and tested on silicon.

Since this register is dynamic it can be slowed down for test purposes by introducing a small capacitor at the output. But since this is a dynamic logic the circuit will malfunction if the slope of the clock is not sufficiently high. Slow clocks cause both the PMOS and the NMOS transistors to be turned on simultaneously, resulting in undefined values of the state and race conditions. The clock slopes should be carefully controlled with the help of local buffers. Also because of the dynamic nature of the circuit the high impedance storage modes make the circuit sensitive to noise and leakage. Often, a feedback transistor is added to the structure (like in the case of DOMINO logic) to make the gate pseudostatic [96].



(a)



(b)

Figure 70: Clock distribution circuitry: (a) Schematic of on–chip clock distribution circuitry, (b) Layout of the the circuit shown in (a). The circuit has been designed such that the propagation delay between clk and clk_n is minimal.

This TSPC D-FF can be used to implement shift registers and counters. The connections

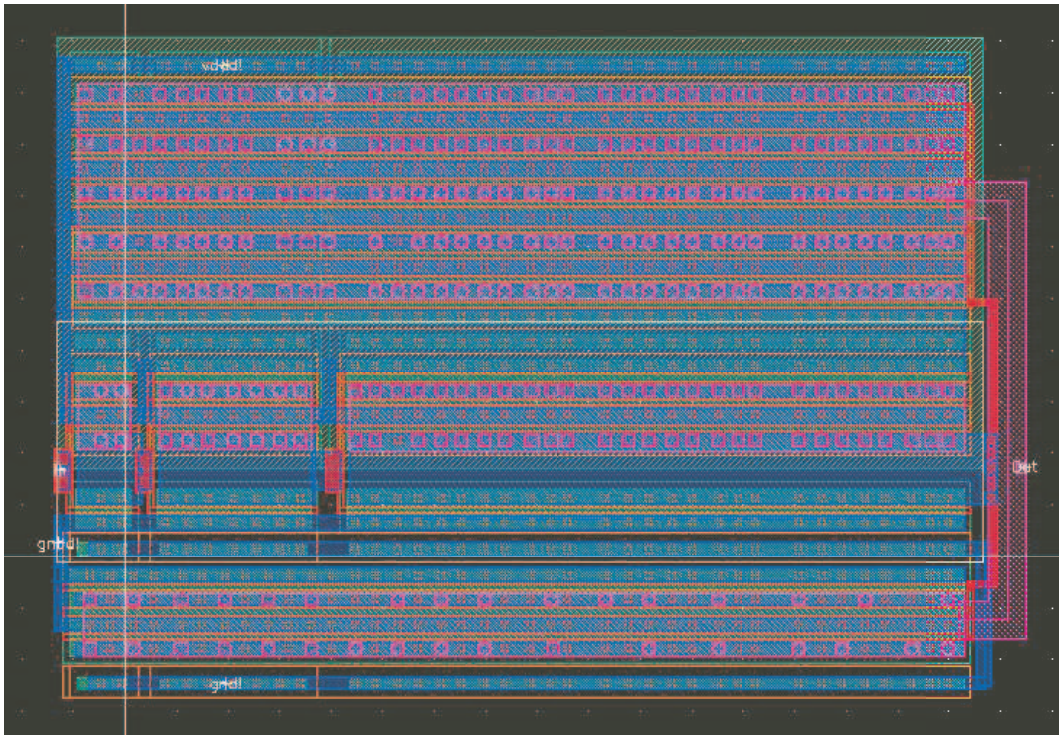
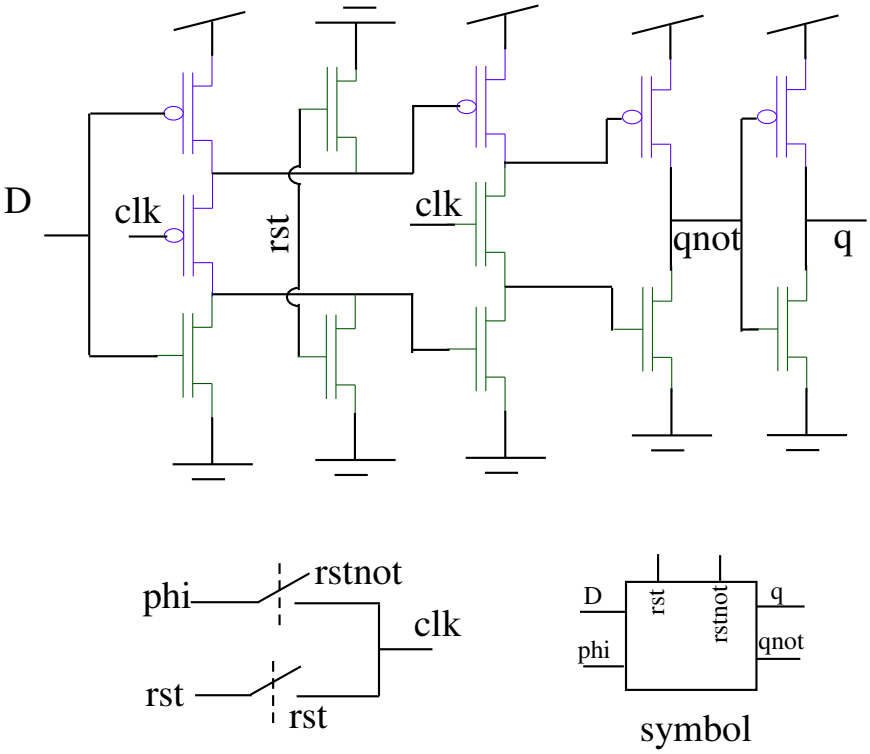


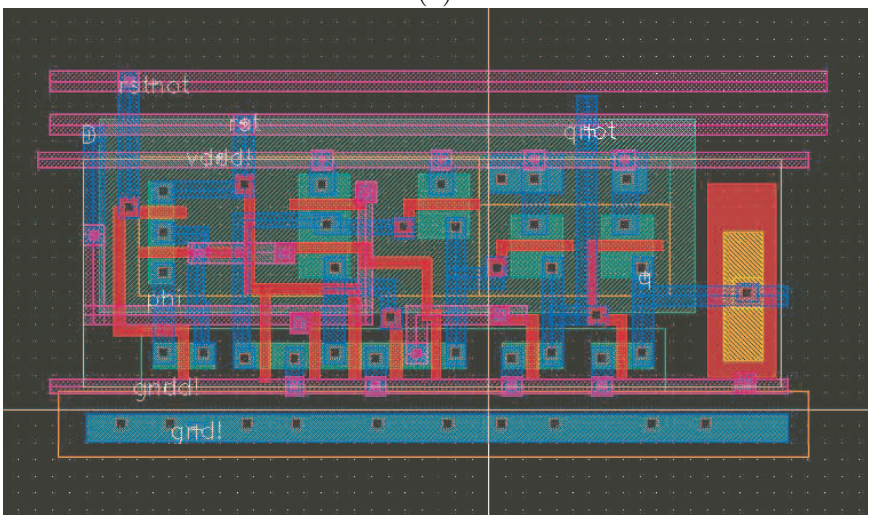
Figure 71: Off–chip driver circuit: A multi–stage buffer circuit was used to drive off–chip components for digital outputs.

required for implementing a shift register and counter are the same as ones as with any D-FF. The shift register circuit was designed and simulated for operations upto 80MHz for a shift register of 10 bits. It was tested for operation uptill 15MHz (limitation of the signal generator). The design can be modified depending on the application for which it will be used for. Similarly a 80 MHz 8-bit counter was designed using this TSPC latch. This was also tested for operation.

I have designed various decoders for the purpose of digital control during FG programming, block selection, and image grabbing. Figure 73 shows a part of the layout for a decoder that is used for programming FG. This decoder has a pitch of 22.5λ. It was simulated (post–layout) for operations upto 80 MHz, and has been tested for operations upto 15 MHz (limitation of the signal generator use for the test). The decoder shown in Figure 74 is the layout of a basic 2x2 decoder that was used extensively for various digital control. I have designed a p-cell such that an arbitrary sized decoders (output bits) can be generated with an user given pitch. It also has drivers at the end so that the slopes of the *output*



(a)



(b)

Figure 72: Flip-Flop using 1 phase clocks with reset: (a) Schematic of Flip-Flop using 1 phase clocks with reset, (b) Layout of the the circuit shown in (a). The circuit has been designed such that they can be tiled in a pitch of 45λ

and $\overline{output}$ are cleaned up, and also the propagation delay between the two are minimized. This was also simulated (post–layout) for operations upto 80 MHz. For further increase in speed other topologies can be chosen, and careful layout should be done to reduce parasitic

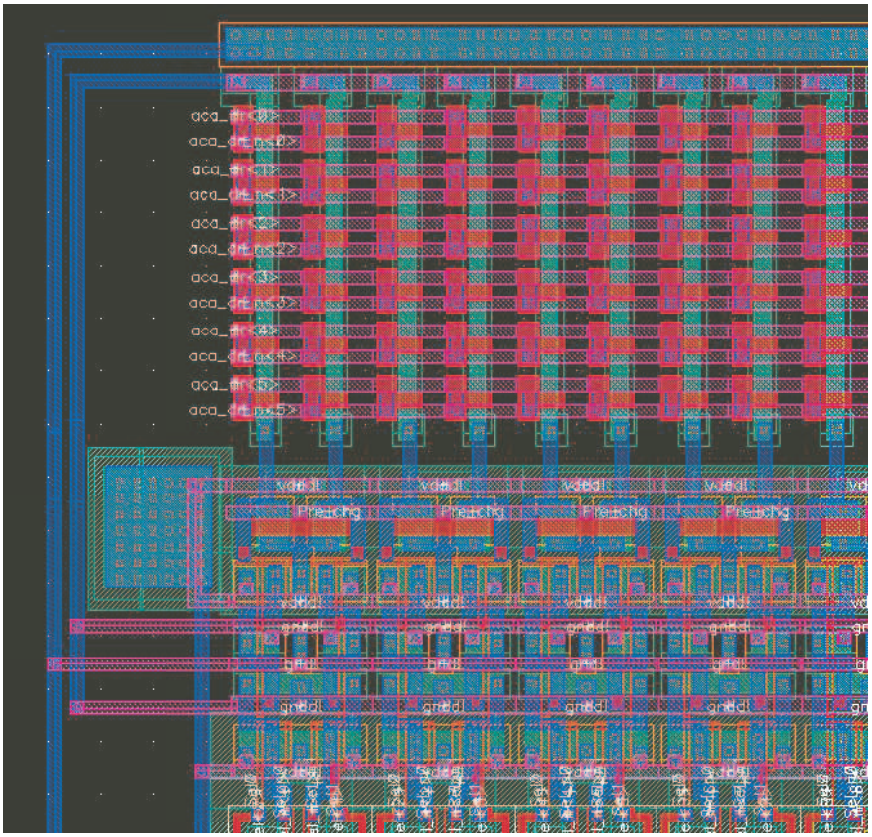


Figure 73: Decoder with a pitch of 22.5λ: A close at the decoder designed for a pitch of 22.5λ. These decoders are used for isolation during programming of floating gates. This decoder was also simulated (post–layout) for operations upto 80 MHz.

effects.

5.3.5 Floor planning and isolation

The chip is a mixed–signal system. There are analog parts and digital parts and they have to be isolated for proper operation. Also because of the way the chip operates there is a lot of interplay between the two. Hence floor planning was important in a system–on–a–chip like this. Care was taken to isolate the analog and digital parts as far as possible given the area I had for layout. The information required to construct a floorplan includes area estimates for each cell and an area estimate of the whole die. Routing area should also be considered during this stage. Figure 75 shows the floor plan for this chip.

Routing was an issue in this chip. Some of the issues that need to be considered are wiring resistance, electron migration, noise coupling, and heat distribution. Since the chip

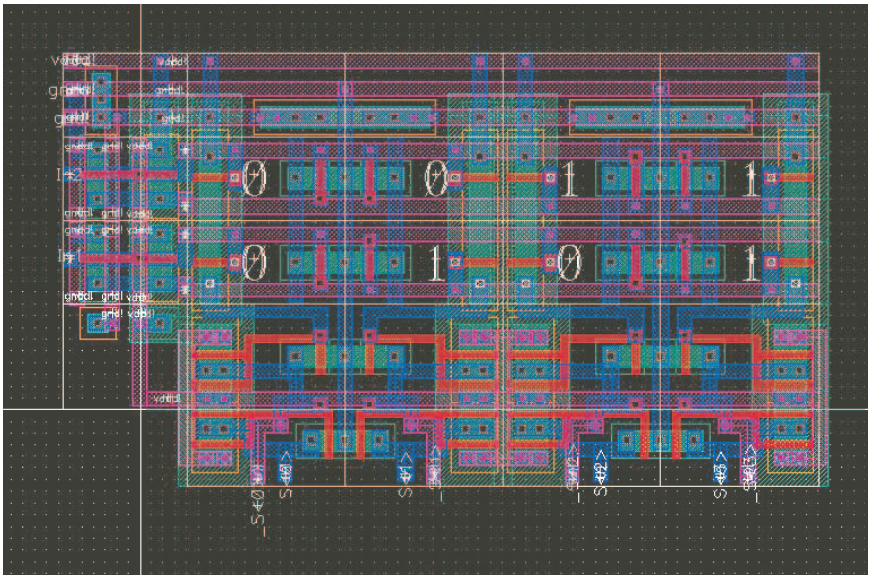


Figure 74: Decoder with variable pitch: A close at the decoder designed for variable pitch. The variables that can be controlled are output pitch and number of digital outputs. This decoder was also simulated (post–layout) for operations upto 80 MHz.

operates in the sub–threshold region the expected current density is not that high hence issues like electronmigration are not an issue. In general care was taken for consideration of the widths of wires for the power lines to minimize metal resistance as much as possible. The minimum width considerations are determined by electronmigration. The minimum lead width can be calculated if the DC currents, maximum allow current density, and the thickness of the metal are known. For the power leads when connecting different metals together, the connections were made through many vias. This is because vias not only increase the resistance of the lead but also limit the amount of current that it can conduct before electron migration causes vias to fail. I have used *channel routing* wherever it was possible as it was quicker to implement and easier to modify. This also minimizes the need for jumpers and thus results in the best utilization of channel space.

Floorplanning also helps to reduce noise. Most of the noise problems encountered in an integrated circuit are caused by capacitive coupling of signals from one circuit to another. A capacitor appears when a lead crosses or runs along another lead. Although the vale of the capacitor itself is very small the amount of energy coupled through them increases with frequency. In MATIA, I was careful not to run digital signals over sensitive analog parts

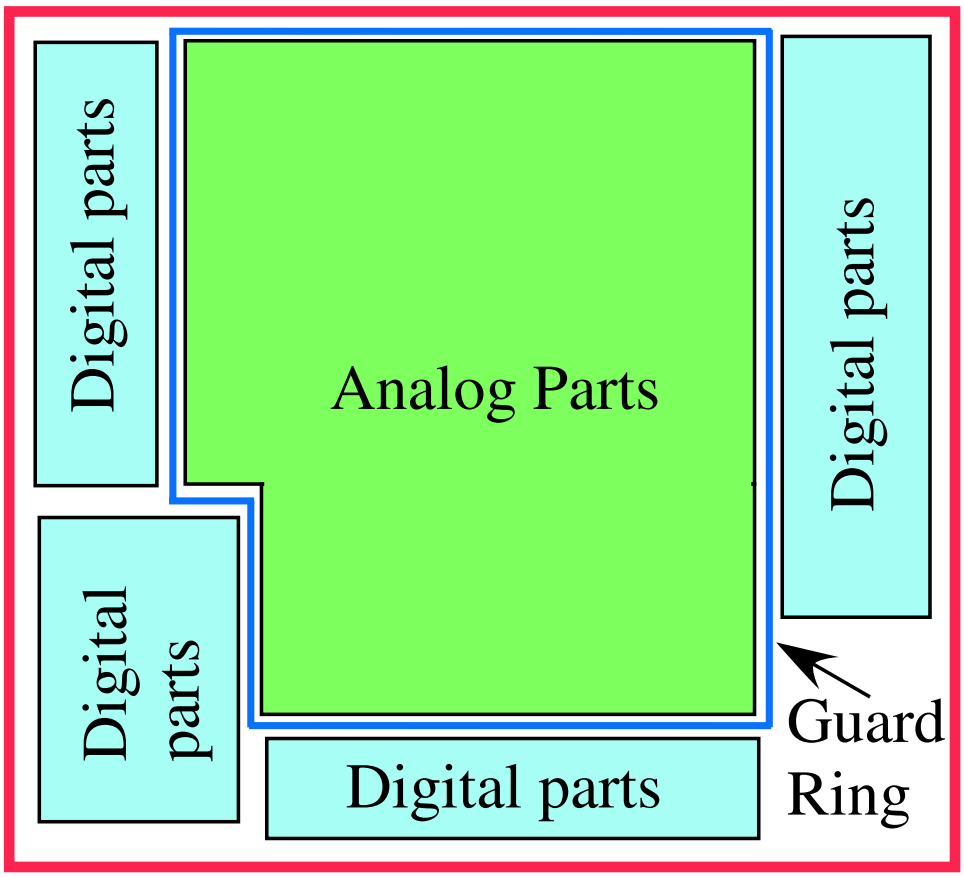


Figure 75: Floor plan of the chip: The floor plan of the chip is illustrated above. The analog parts consist of the pixel array, the FG bias generators, FG VMM and other bias circuits. The digital parts consist of various decoders, multiplexors, clock distribution circuitry and digital buffers.

of wires carrying sensitive signals. If I had to run something over a sensitive part then I shielded that part using a metal layer in between which was held a fixed potential (ground). This layout method reduces the cross–talk effects. It, however, is obtained at the cost of more complex wiring and greater capacitance between the signals and ground. Since the architecture is differential most of the cross–talk is reduced to common–mode disturbance. Hence cross–talk is much reduced in this system. In low noise applications, long signal wires, with sheet resistance of 40 to 80 m$\Omega/\square$, may introduce substantial thermal noise [97].

Since this is a mixed signal chip there are a number of references that have to be provided. These were distributed in the *current domain* rather than voltage domain. The idea is to route the reference current to the vicinity of the building block and then perform

the current mirror operation locally. Using this method reduces the systematic mismatch due to the voltage drop along the ground line.

Substrate coupling is an important issue as CMOS technologies use a heavily–doped p^+ substrate to minimize latchup susceptibility. However, unwanted paths are created between various device in the circuit due to the low resistivity of the substrate. Since the architecture is differential the circuits are less sensitive to the common–mode noise. I have also distributed the clocks in complementary form to reduce the net coupling noise. I have also used guard rings to isolate the sensitive sections from the substrate noise produced by other sections. A guard ring is usually a continuous ring made of substrate ties that surround the circuit. It provides a low–impedance path to ground for the charge carriers produced in the circuit. An n-well ring helps too as with it's large depth it stops noise currents flowing near the surface. I have also connected ground and substrate on–chip and brought out through a single wire. The substrate is connected to the analog gnd. If these have unequal bounce then the drain currents are corrupted by substrate noise. I have also separated the analog and digital grounds and also have separate analog and digital supplies. These are connected off-chip at only one point. If pin limited then I connect them on–chip at only one place right after the supply pads.

All the above mentioned layout techniques and floor planning have significantly improved the quality of images that can be read using the MATIA.

5.4 Flash structure for video processing

There has been an increasing demand for low–cost, and high–speed integrated analog–to–digital converter (ADC), in the field of video signal processing. The potential of flash architectures for realizing high resolution and fast ADCs have been demonstrated in a number of deigns [12, 98, 46]. However, these designs require high speed sample and hold amplifier, a high precision ADC, and have large power dissipation. These factors along with space constraints prevent conventional ADCs to be incorporated into large imaging architectures. Integration of analog and digital components in system–on–chips also pose problems for voltage-mode flash ADCs. Voltage mode ADCs also require precise resistors

and capacitors, which need fabrication steps that might not be provided by many digital processes. The high speed and low power constraints can be realized by using current–mode techniques. Current–mode circuits provide advantages like immunity from deleterious influences like ground and power supply noise, and signal line impedance [110]. These circuit techniques have also been used for high-speed applications [12, 73].

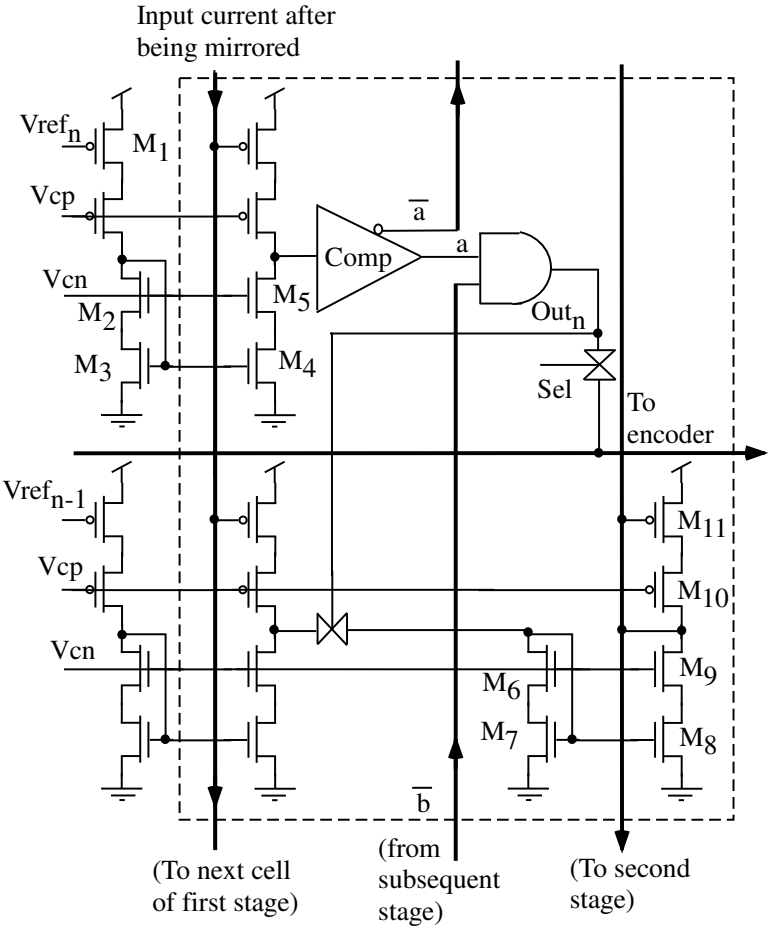
We propose a novel current-mode cell that can be used in one or two-step flash ADCs for video applications. This cell uses floating-gate technology to store the reference currents on–chip. The structure is programmable and thus can be fine-tuned for better performance. The cells have been implemented such that they can be tiled for parallel readout and fit within a pitch of 13.5 μm.

Section 5.4.1 gives an overview of the ADC cells. In section 5.4.2 the current–mode comparators used for the cells have been described. Section 5.4.3 deals with the floating gate current reference circuits. The results are presented in section 5.4.4.

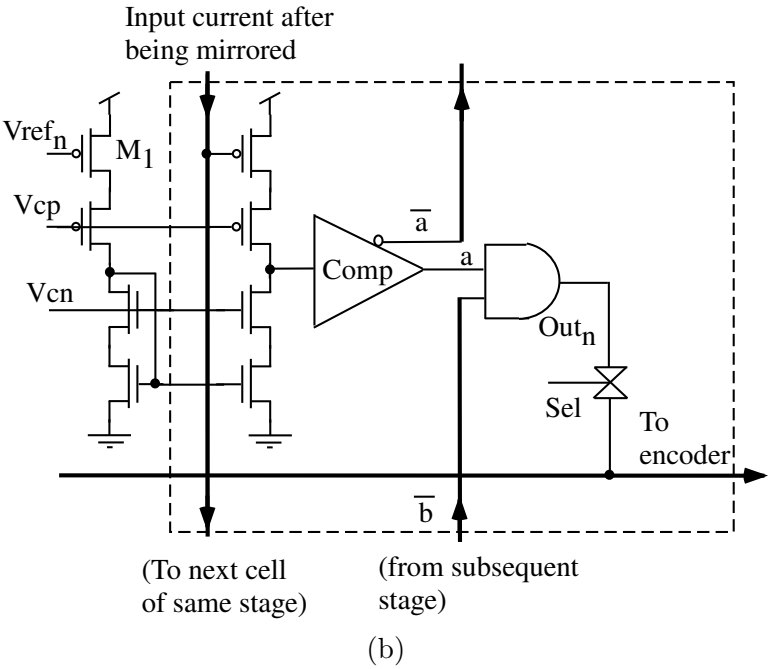
5.4.1 System overview

In this section two cells are proposed that can be used for building two-step and one-step flash ADCs. Floating gates technology enables the reference currents to be stored on–chip.

Figure 76(a) shows the proposed cell for the first stage of a two–step flash. The input current (I_{in}) is mirrored using PMOS transistors and is transmitted to all the stages. For each stage this current is compared to the reference current (I_{ref}) for that stage. This reference current is being generated by a floating gate transistor (M_1). For comparison with the input current this reference current is mirrored using nmos transistors (M_2-M_5). The comparison is done using the comparator structures described in section 5.4.2. The output is high if the input current is less than the reference current and is low for the opposite case. The output of this comparison (OUT_n) is then AND-ed with the inverted output of the next stage ($\overline{b}$) to generate the thermometer output for this cell. If the thermometer output is high then a fixed bias current (I_{ref-1}) is subtracted from the input current. This residual current is then mirrored (M_6-M_9) and is transmitted as the input to the second-stage of the two-step flash architecture (M_{10} and M_{11}). The operations can be summarized as follows:



(a)



(b)

Figure 76: Architectural blocks for flash ADCs: (a) Cell for the first stage of a two–step flash ADC, (b) Cell for a one–step flash or the second stage of a two–step flash ADC.

IF $(I_{in}) < (I_{ref})$

THEN a is HIGH;

ELSE IF $(I_{in}) > (I_{ref})$

THEN a is LOW;

IF (a AND $\overline{b}$ = HIGH)

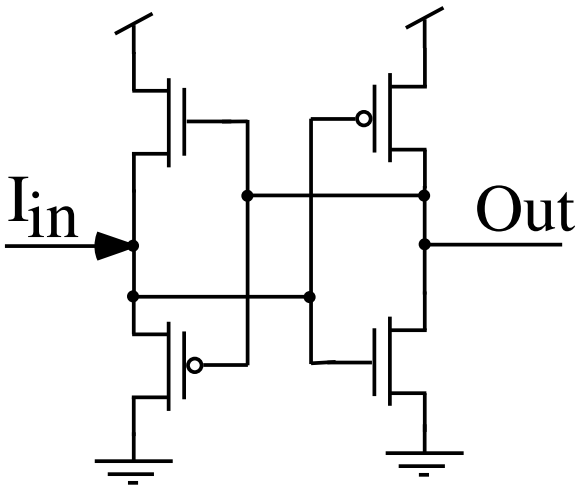
THEN I_{ref-1} is subtracted from I_{in};

For the second-stage of the two-step flash architecture or for implementing a one–step flash, the cell described in Figure 76(b) can be used. In this cell the input current (I_{in}) is compared to the reference current for that cell (I_{ref}). Like the first cell the reference current is generated by a floating gate transistor (M_1). The output, for this cell, is high if the input current is less than the reference current and is low for the opposite case. For generation of the thermometer code this output is AND-ed with the inverted output of the next stage ($\overline{b}$). The output is then placed on to the encoder depending on the signal *Sel*. The *Sel* signal is used to multiplex the outputs of the different ADC cells when these would be tiled for parallel operation.

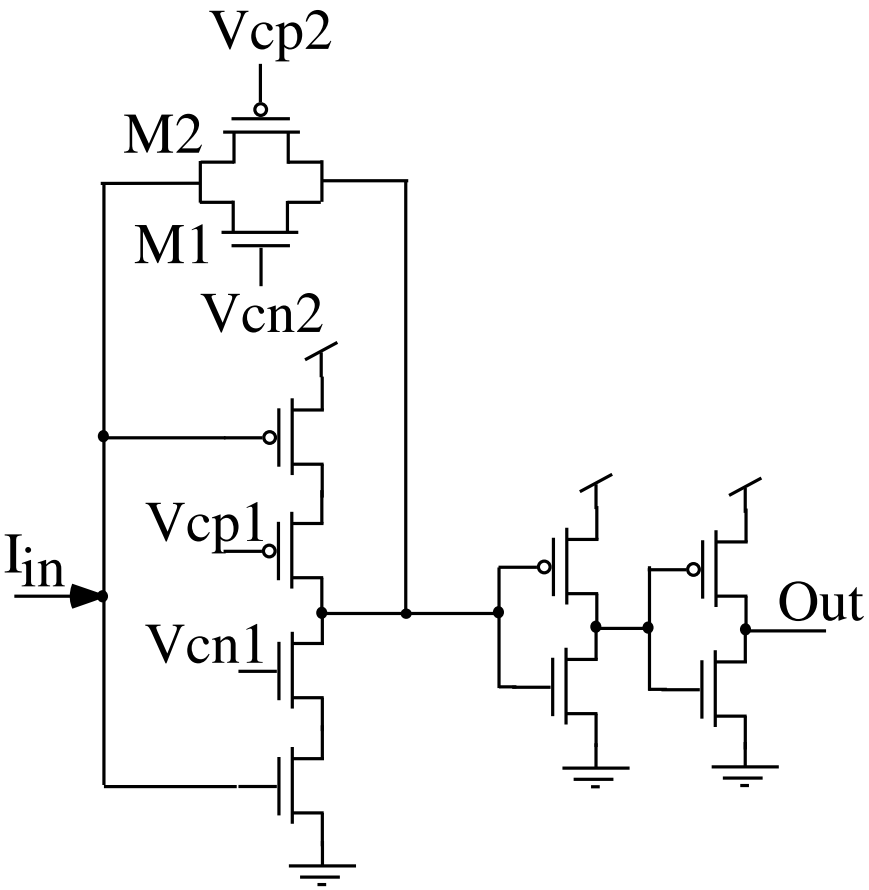
5.4.2 Current mode comparators

Since the expected current input is in the lower nano-amperes range a comparator with very high accuracy and high speed for low currents has to be used. Figure 77(a) shows the current comparator that was proposed in [111]. This comparator uses a source follower as the input stage and a CMOS inverter as the positive feedback, which enables faster response time. The dynamic response of this comparator, for small input currents, suffers because there exists a deadband region in which the two input transistors are temporarily turned off. Since the input resistance increases during this time the dynamic response time for smaller input currents increases.

We use an asynchronous mode current comparator that uses non-linear feedback to combine the advantages of capacitive–input and resistive–input [27] as shown in Figure 77(b). In order to achieve faster response times for the current switch comparators input and output nodes have been uncoupled. Simple inverters have been used as amplifiers



(a)



(b)

Figure 77: Current mode comparators: (a) Conventional CMOS comparators [111], (b) Current comparators which employs nonlinear feedback to obtain high accuracy and high speed for low input currents [27].

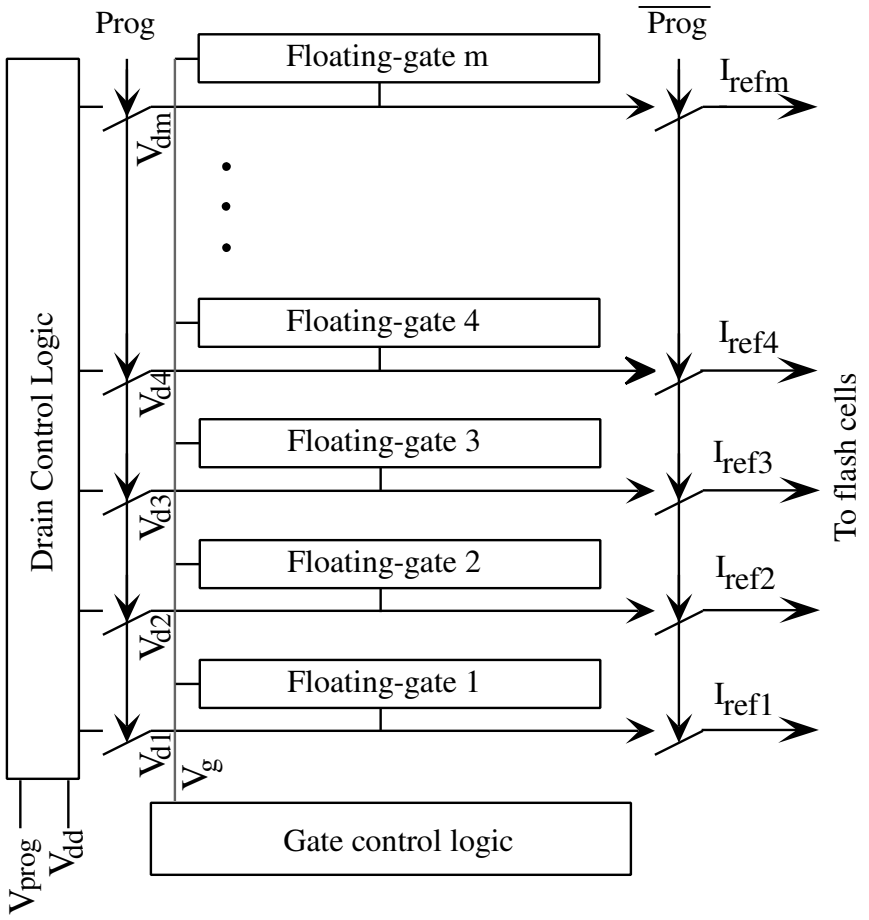


Figure 78: Top-level view of our reference generation circuitry: (a) In operation (transform) mode we have an array of stored values that are output in sequence. In programming mode we can easily reconfigure this circuitry on the outside edges for programming.

to increase speed. Cascode transistors have been used at the output node to reduce the effect of the overlap capacitances. The input current is integrated at the input capacitance. For positive currents the input voltage continuously increases. the amplifier causes output voltage to decease faster than the input voltage. This causes V_{gs} of M_1 to increase while decreasing V_{gs} of M_2. M_1 in driven into on state and a negative feedback loop is created around the amplifier. For negative currents a similar situation occurs with M_2 providing the negative feedback.

5.4.3 Floating–gate reference circuits

Floating–gate circuit elements are used to store and generate currents needed for the various references for each ADC cell. This section deals with the programming of floating–gate

arrays.

Figure 78 shows the top–level view of our reference generation circuit. In programming mode, we can easily reconfigure this circuitry on the outside edges for programming. This approach is compatible with our standard programming structure and algorithm. Each floating–gate can be isolated for programming using the peripheral digital control circuits.

The array of floating–gates are initially tunnelled so that they have almost negligible currents for the operating gate voltage. The tunnelling voltage required for this process is 15 V. They are then programmed, using hot electron injection, to the various currents needed for the reference generation. The floating–gates can be used to store arbitrary waveforms. The floating–gates are programmed so that they supply different currents for the same bias gate voltage. In operation (transform) mode these currents (I_{ref}) are used as trip points for the current–mode comparators.

5.4.4 Characterization of the flash structure

The cells were fabricated using 0.5μm N-well CMOS technology. They were simulated, from the extracted values, using SpectreS. The cells have a pitch of 13.5μ. The cell described by Figure 76(a) has a size of 13.5μm x 122.7μm, while the one in Figure 76(b) has a size of 13.5μm x 87.15μm. They were designed such that they can be tiled together for parallel readout. These cells can be used for current readout from imagers or any other structure that requires fast parallel readout and has current outputs. These cells were designed to operate for small current inputs.

Figure 79 shows that accurate current trip points can be stored on–chip. For Figure 79(a), 256 equidistant current reference values have been programmed between 150pA and 40nA. These can be the trip points for a 8–bit one–step flash ADC using these cells. Figure 79(b) shows the values of the first sixteen values. These can be reference values for the second stage of a two–stage flash structure. Since we can program any reference current values on–chip we can operate the flash cells over various ranges. We can also change the trip points as necessary for better SNR and performance.

The transient response of the comparators were tested for input square–wave currents

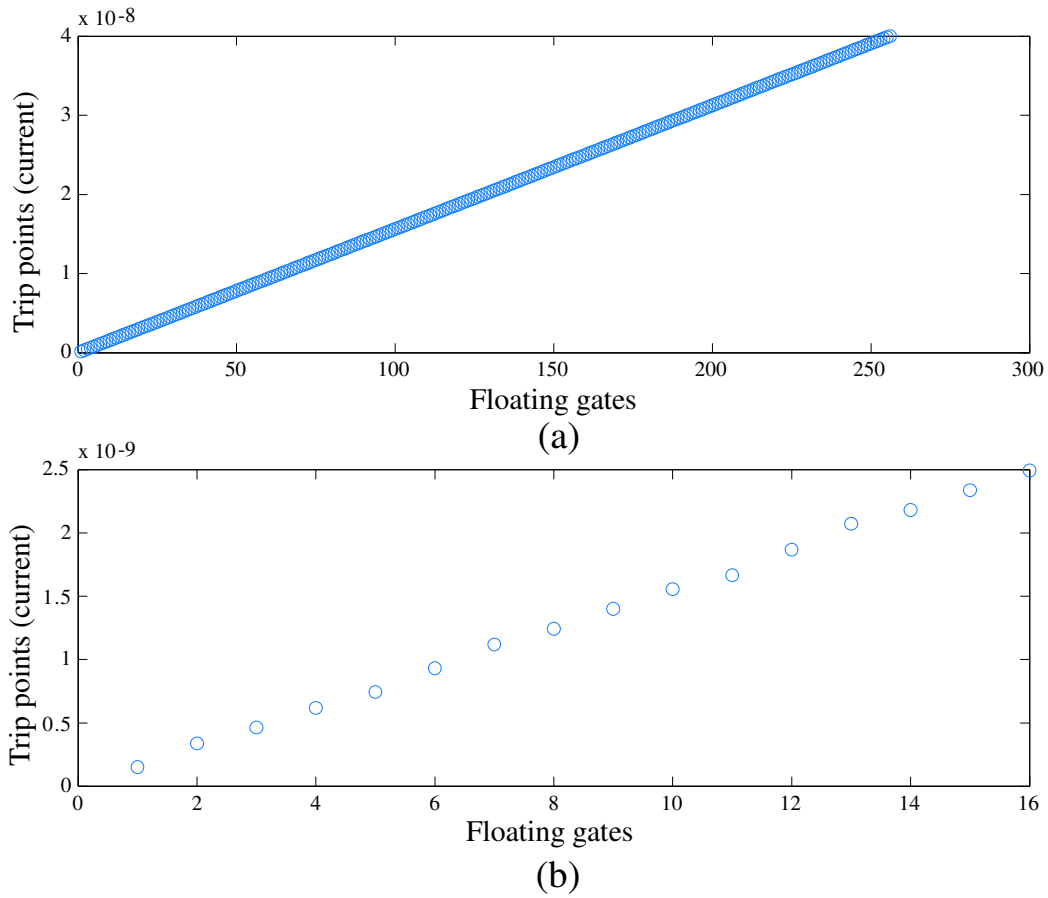


Figure 79: Programming reference currents on–chip: (a) Plot showing programmed values of reference currents. These are the trip points for a 8–bit one stage flash structure. (b) Plot showing that accurate programming can be achieved for on-chip reference currents.

with amplitudes ranging upto $\pm 50\mu$A. Figure 80 shows the propagation delay for the different comparators for various differential current inputs. The comparator in Figure 77(b) has a faster response for lower currents, as expected. For higher currents the propagation delays are almost same. For that comparator 20pA of differential input current induces 3.7μs of propagation delay, while for 50μA of differential input current the propagation delay is 2ns.

If an ADC is designed using these structures it can be observed that if the differential currents are in the order of tens of micro–amperes then operations in the vicinity of 125 MSamples/sec are possible with careful design of the additional circuitry. It is also observed that for small currents, in the range of tens of pico–amperes, the overall ADC can operate at rates of 100 KSamples/sec. Figure 81 shows the layout of a test chip that was fabricated

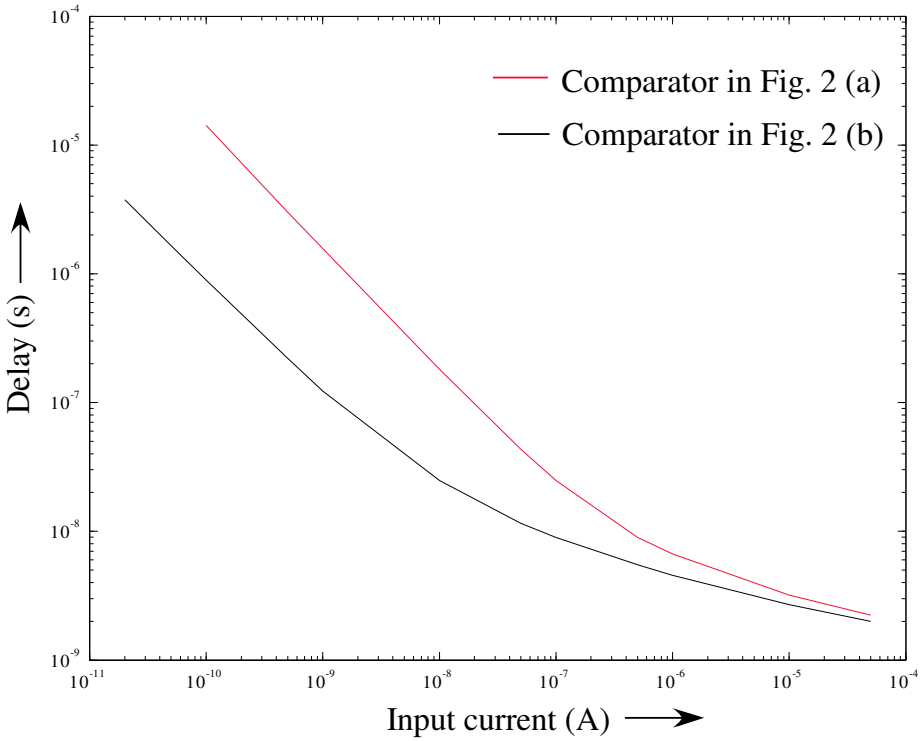


Figure 80: Comparison of different comparators: Plot of propagation time versus differential input currents for the comparators described in Figure 77.

using 0.5μm Nwell CMOS.

5.5 PCB and timing sequence

Figure 82 shows the schematic of the printed circuit board used to control the imager. The analog biases (drain voltage, gate voltage, tunneling voltage, and power supply) required by the imager chip are provided by octal-DACs which are controlled by the FPGA. There are additional circuitry for generating the tunnel voltage (15 V), and all the analog voltages are buffered before they are presented to the imager chip. Since a high power supply (6V–6.5V) is required for programming, the digital signals to be used during that phase go through level shifters. The board also has 14-bit 10 MS/s ADCs for image capture. We test our imagers by projecting a directed light source on our imager through a complex lens system. During our experiments we have seen little noticeable movement of the floating gate elements from their respective programmed values. The PCB is a four layered board with separate supply and ground planes, and was designed using EAGLE^{TM}.

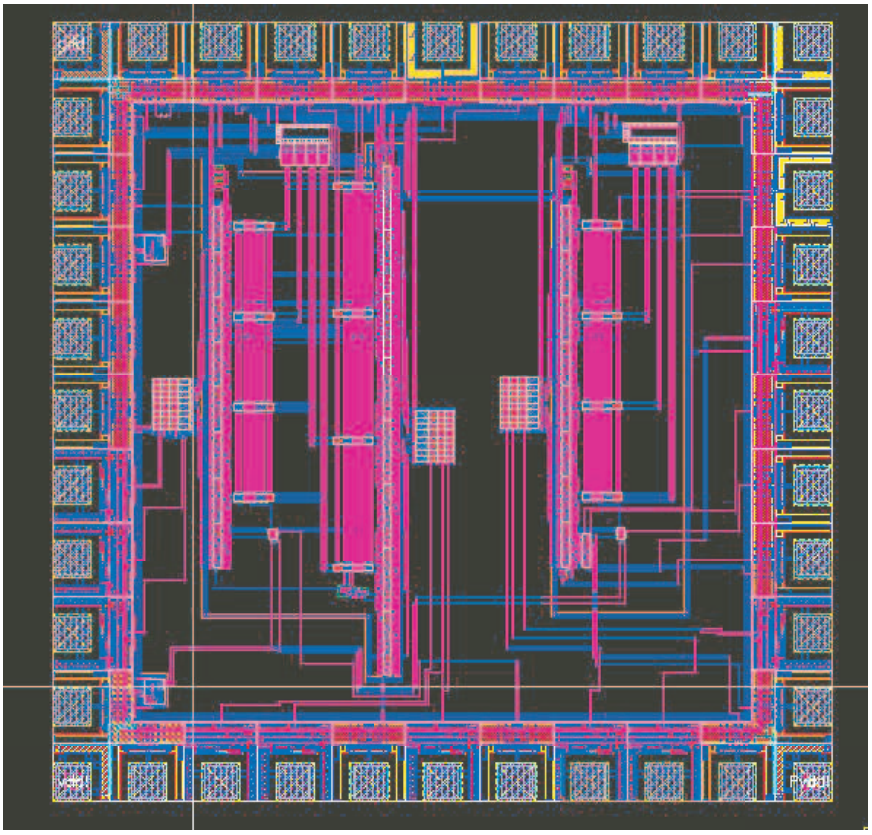


Figure 81: Flash test chip: The flash test chip consists of a 3 bit flash and a 8 bit two step flash. Bubble suppression and decoders were also designed for proper read–out.

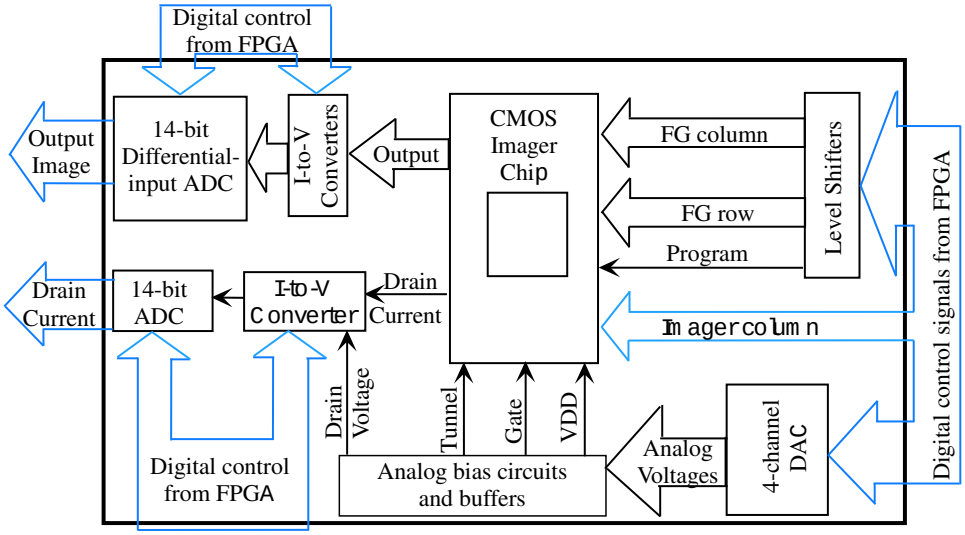


Figure 82: Schematic of the printed circuit board (PCB): A PCB was designed for providing the analog biases and the digital control required for the chip. All the digital control are provided by a field programmable gate array (FPGA). The digital outputs are acquired and stored using the FPGA for further processing.

Table 3: Table of parameters and results for the current-input comparator cell for a Flash ADC array.

Process	0.5μm Nwell CMOS
Supply	3.3V
Cell area	13.5μm x 87.15μm (1 stage–Fig.76a) 13.5μm x 122.7μm (2 stage–Fig.76b)
Propagation Delay I_{in} = 20pA I_{in} = 50μA	 3.7μs 2ns
Tunneling Voltage	15V - 18V
Injection V_{dd}	5V - 6.5V

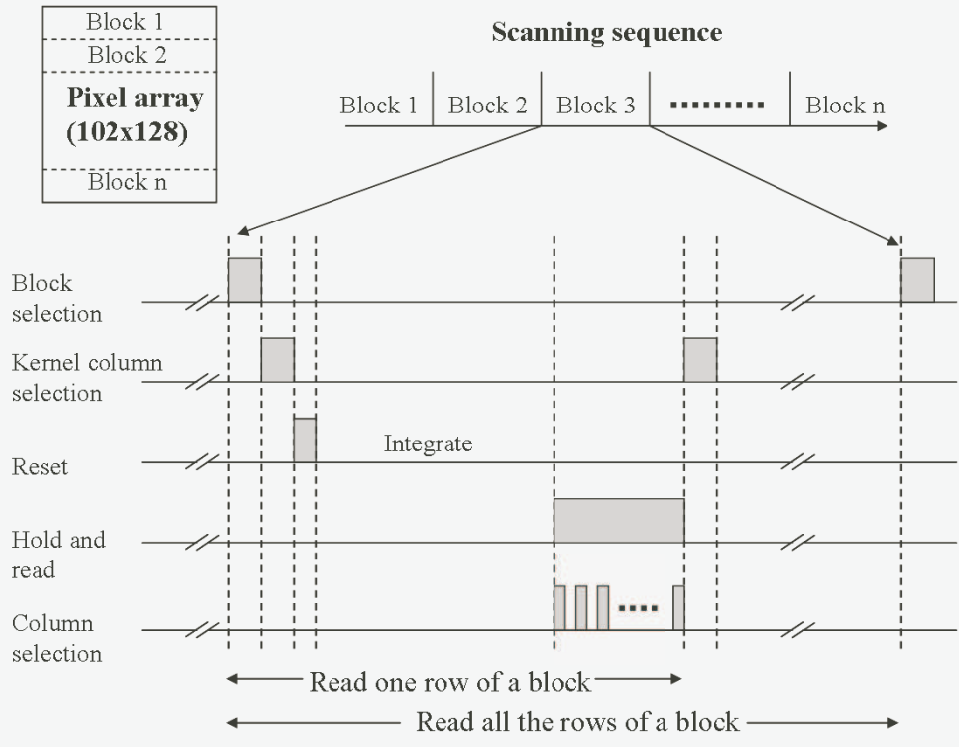


Figure 83: Timing sequence for image readout: The images are readout in a column parallel fashion. Random access is possible for reading parts of the image. The timing sequence is provided using a FPGA.

The FPGAs used are Altera Stratix and Xilinx Virtex-E. The VHDL modules for controlling the different modules in the board were controlled using embedded microprocessors, Nios and Microblazer, respectively. This was in turn controlled by MATLAB^{TM}. The test setup was configured such that it can be fully controlled through MATLAB^{TM}. The data was transferred from the FPGA to the computer through an ethernet connection.

Figure 83 shows the timing sequence for image readout. The blocks are selected first and the different rows of the basis function are presented to the chosen block using kernel column selection. The outputs are readout using column parallel I-Vs and open loop sample and hold circuits. After all the rows of the basis function have been presented for in-pixel multiplication and the outputs have been readout, the block selection is changed, and the above process is repeated for reading that block.

5.6 On-chip image transforms

I have built several functional imagers in 0.5μm CMOS technology of sizes 16x16, 48x40, and 128x128. The 128×128 size imager uses a block transform window of 16, therefore requires an array of 252x16 floating gates to store the required basis functions. All of these systems contain the necessary control circuits that allow for programming of individual floating gates. We program the floating gate elements to arbitrary values using an external programming board that only requires an external power supply and field programmable gate array FPGA interface.

Figure 84 shows the effects of different transforms on an image. These results are from a 16×16 imager block. Figure 84(a) shows the image of a line, while Fig. 84(b) shows the smoothed image obtained from a 2×2 low pass filter kernel. As expected some edge information is lost and the speckle–type noise, as seen in Fig. 84(a), has been smoothed out. Figure 84(c) shows that thresholding an high–boost filtered [48] image leads to edge detection. The thresholding was done using using MATLAB^{TM} but it would be easy to implement on–chip. Figures 84(d–f) show the block DST, DCT type -II, and Walsh-Hadamard transforms of the image, respectively. A common characteristic of these images is that their magnitude responses have most of their energy concentrated in the low frequency regions of the plot. MATIA can be configured to read an image as shown in Fig. 85. Figure 85(a) shows the high resolution original image that was placed on the MATIA. Figure 85(b) shows the image of Fig. 85(a) subsampled to 48x40 using MATLAB^{TM}, wile Fig. 85(d) shows the output of a 48x40 MATIA when it was configured to read an image. There was no averaging of frames performed for this image and the quality can be improve by averaging

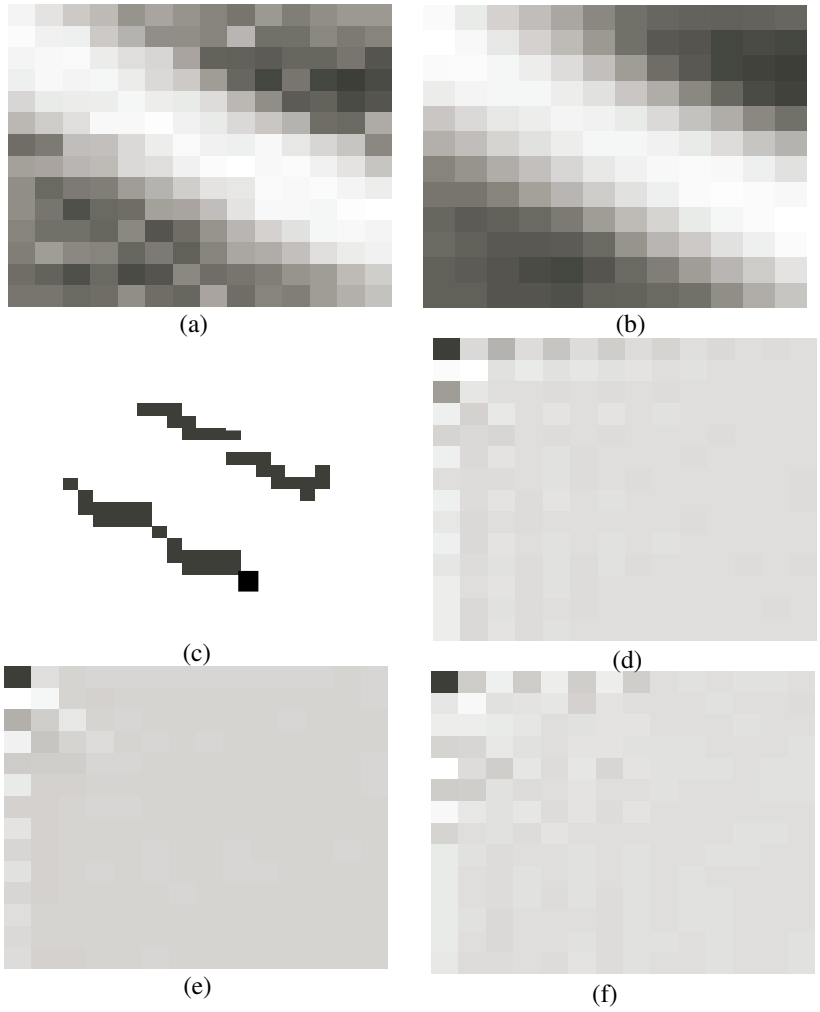


Figure 84: Different transforms on a 16×16 imager block: (a) Image of a line; (b) Smoothed image obtained from a 2×2 low pass filter; (c) Edge detection after high boost filter and thresholding; (d) Result of a block DST transform; (e) Result of a block DCT type -II transform; (f)Result of a block Walsh-Hadamard transform.

over several frames. Figure 85(c) shows the smoothed image (2x2 low pass filtered version of Fig. 85(b)) performed using MATLAB^{TM}, while Fig. 85(e) shows the output of the 48x40 MATIA when the smoothing was performed on–chip. Different sizes smoothing kernels can be programmed onto the MATIA and convolution can be performed.

A transform imager is capable of computing block transforms. Due to its programmability, we can implement various block transform such as DCT, DST, Hardamard transform,

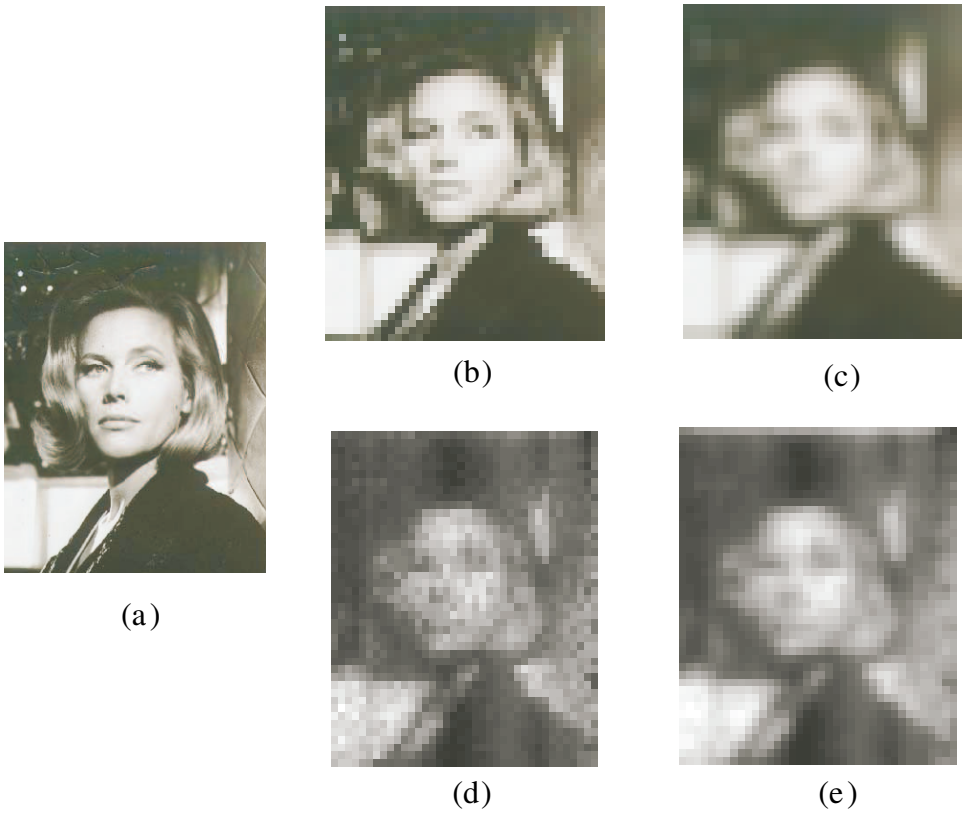


Figure 85: Images from 48x40 MATIA: (a) High resolution original image, (b) Original image sub–sampled to 48x40, using MATLAB^{TM}, (c) Smoothed version of (b) using a 2x2 filter using MATLAB^{TM}, (d) Image from MATIA when it was configured to read an image, (e) Smoothed image (2x2 low pass filter) using MATIA

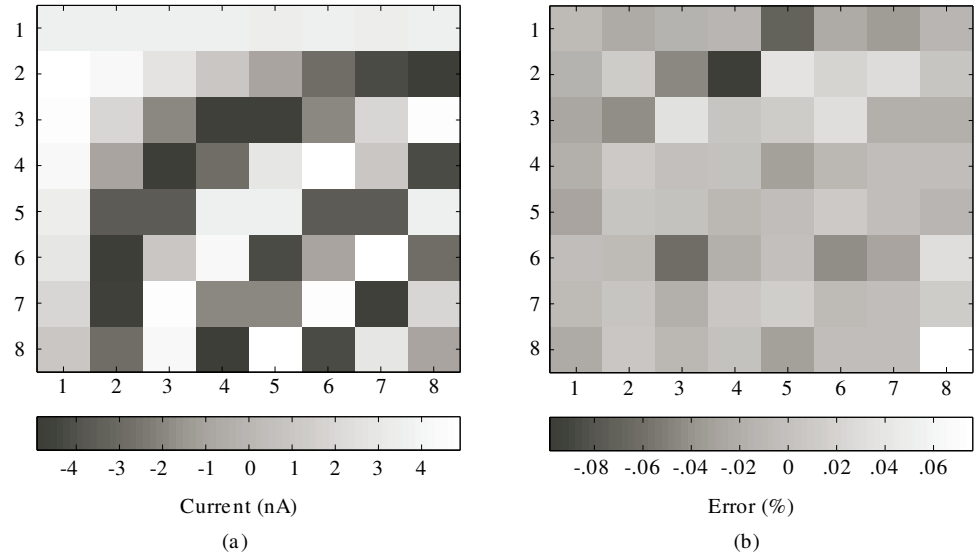


Figure 86: Programmed DCT values for JPEG compression using MATIA: A 8x8 DCT kernel was programmed onto a 8x8 array. The values were programmed around a DC of 10nA. The DC was subtracted for clarity of display. The maximum percentage deviation for the array was 0.07%.

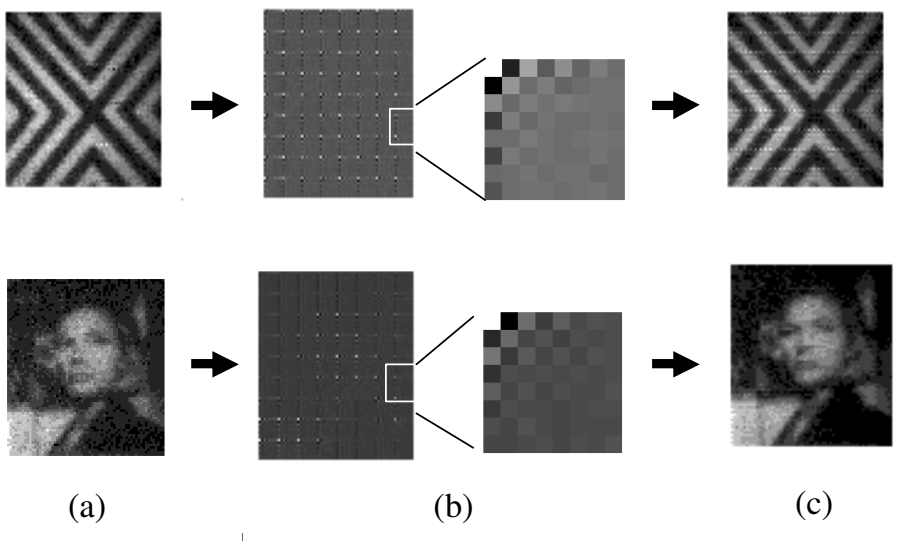


Figure 87: Discrete cosine transforms (DCT): Floating gates can be accurately programmed to any desired current for any set gate voltage. The above plots show the output of two rows (each of 16 elements) that have been programmed to differential sinusoidal current waveforms and differential triangular current waveforms, respectively.

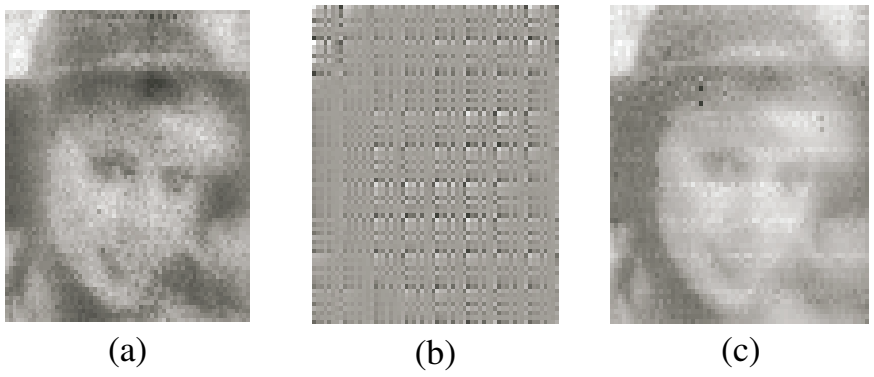


Figure 88: Haar Transforms: Floating gates can be accurately programmed to any desired current for any set gate voltage. The above plots show the output of two rows (each of 16 elements) that have been programmed to differential sinusoidal current waveforms and differential triangular current waveforms, respectively.

Haar transform, and so on. We show DCT and Haar transform as examples because DCT is a well known useful block transform for years, and Haar transform is used for wavelet-based compression, which is adopted in JPEG2000.

For implementing DCT the DCT coefficients had to be programmed onto the MATIA. Figure 86 shows the programmed DCT values and the corresponding programming error.

The error seems to be random and is the case for every experiment. Figure 87 shows the original image as acquired by the transform imager, the 8x8 block DCT output of the imager, and the corresponding reconstructed image without compression. The reconstructed images from the DCT coefficients show that the computation process has introduced minimal sources of error. The difference between reconstruction from the compressed image and reconstruction starting from the original image is negligible. Figure 88 shows the Haar transform results. We programmed the transform imager with 8x8 Haar transform. Figure 88(b) shows Haar transform before reordering, and Figure 88(c) is a reconstructed image to verify the operation of the chip. In all the above experiments no averaging over frames was performed. We have observed that averaging over frames enhances the image quality, but as can be observed from the presented data even for single frames we get 'clean' transforms. We have computed these images and transforms up to 25 frame per second (fps). We have characterized our single-column pixel reading speed and found we could achieve greater than 60 fps for one Mega pixel imager.

5.7 Low–power baseline JPEG

It has been recognized for several decades that block transforms represent attractive transform operatives for image coding. There are many unitary image block transforms that have the property of packing signal energy at the beginning of the transformed image, and thus, image compression algorithms can be built around this idea. JPEG, which stands for Joint Photographic Experts Group, is the accepted block transform coding standard. It is the official algorithm of the International Standard Organization (ISO) for image compression, adopted in 1992. The JPEG algorithm has four modes: baseline-sequential, progressive, lossless and hierarchial. We focus on the baseline system. Figure 89 shows the block diagrams of the various methods in which JPEG can be implemented. Figure 89(a) shows the conventional implementation where all the processing is performed in the digital domain; Figure 89(b) shows present implementations using MATIA, while Figure 89(c) shows a single chip implementation of the whole system. The image transform coding strategy can be described as follows [102]:

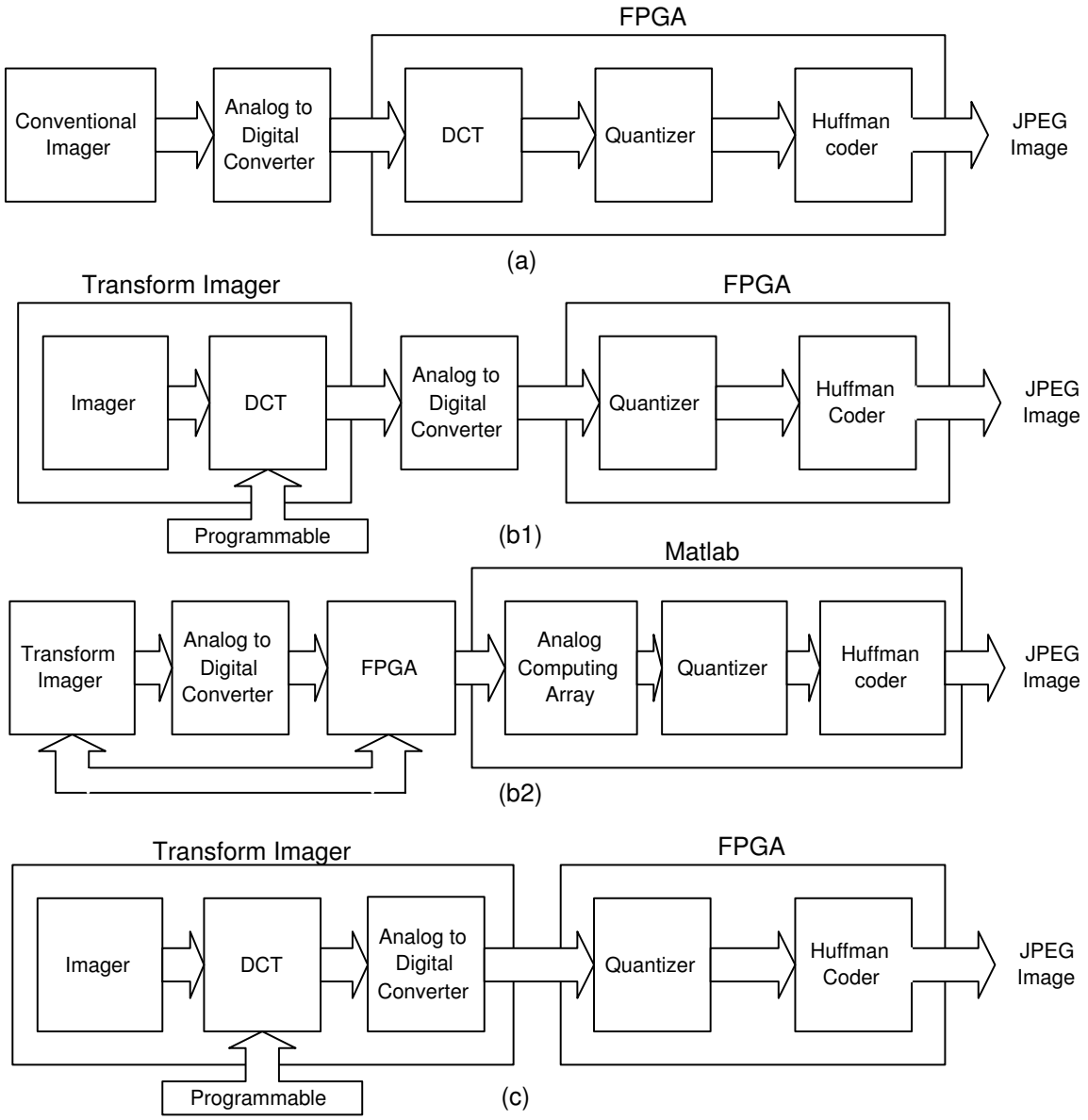


Figure 89: Block diagram of JPEG algorithm using MATIA: Top level view of our JPEG system used as an application for signal processing. (a) Conventional approach (b) Our proposed system systems. MATIA will be used to compute the DCT transform. FPGA would be used for encoding. (c) Ideal single chip system.

- Partition the image into contiguous blocks. By convention these blocks are typically square N×N blocks where N = 8. Square partitionings of this sort assume inherently that the input image dimensions are multiples of N. Alternately, circular or symmetric extensions can be employed.

- Compute the DCT of each block. The DCT has become the overwhelming favorite for

image compression because it can be implemented very efficiently and its performance for natural images is the best of the candidates available.

- Quantize the block transform coefficients. This implies that some attention is given to the number of quantization levels that should be allotted to each block and what type of quantizer should be used.

- Entropy code the transform coefficients. The first coefficient in each block is the DC component and is typically larger in amplitude compared to the AC values. The DC coefficients are coded by first computing the first backward difference and then using a DC Huffman table. The AC coefficients are all coded on a block–by–block basis. *Runlength coding* is used to code the AC elements. The baseline JPEG is considered to be "sequential" because the blocks are processed and coded in a scanned order, starting from the upper left block and terminating with the lower right.

For JPEG the DCT-II equations are as follows:

$$X[k_1, k_2] = \frac{1}{4} \sum_{n_1=0}^{7} \sum_{n_2=0}^{7} C_{k_1} C_{k_2} x[n_1, n_2] cos[\frac{(2n_1 + 1)k_1\pi}{16}] cos[\frac{(2n_2 + 1)k_1\pi}{16}] \tag{52}$$

with inverse as:

$$x[n_1, n_2] = \frac{1}{4} \sum_{k_1=0}^{7} \sum_{k_2=0}^{7} C_{k_1} C_{k_2} X[k_1, k_2] cos[\frac{(2n_1 + 1)k_1\pi}{16}] cos[\frac{(2n_2 + 1)k_1\pi}{16}] \tag{53}$$

where,

$$C_{k_1}, C_{k_2} = \{ \begin{array}{ll} \frac{1}{\sqrt{2}} & for \quad k_1, k_2 = 0 \\ 1 & otherwise \end{array} \tag{54}$$

The Q matrix is given below:

Figure 90: JPEG Compression: Floating gates can be accurately programmed to any desired current for any set gate voltage. The above plots show the output of two rows (each of 16 elements) that have been programmed to differential sinusoidal current waveforms and differential triangular current waveforms, respectively.

$$A_2 = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix} \tag{55}$$

As can be observed that the values in the low–frequency region are small compared with the values in the high–frequency region. This gives preference to the dominant low frequencies. The precise amount of compression can be controlled by scaling the Q–matrix.

Our architecture can be used as a JPEG encoder. The block DCT is computed on–chip. The output can then be encoded using a FPGA. Since most of the digital coding involves looking up the corresponding code from a look–up table this can be easily performed on the FPGA.

Figure 90 gives the peak signal to noise ratio (PSNR) of the images with different compression ratios. The following PSNR formula was used.

$$PSNR = 20 \log \frac{1}{\sqrt{MSE}} \tag{56}$$

Table 4: Comparison of JPEG implementations

Implementation	**FPGA-only**	**Proposed**
Number of slices	1751	519
Number of slice flip flops	2615	467
Number of 4 input LUTs	799	35
Number of BRAMS	2	0
Total power estimation	183mW	37mW

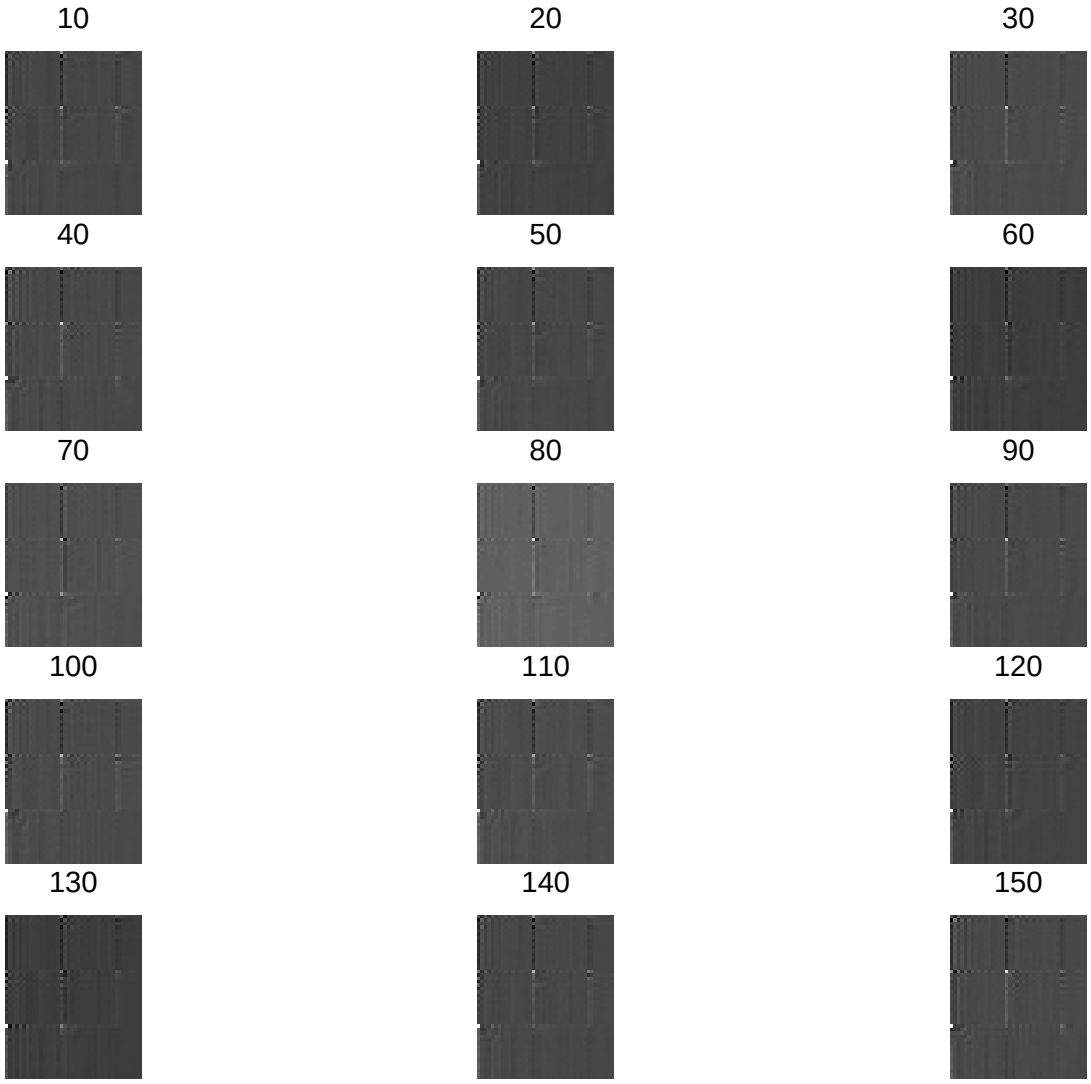


Figure 91: Motion JPEG Using MATIA: Output of MATIA system when it was configured to compute DCT of each frame at 20fps. The numbers indicate the frame number.

where images are normalized to [0,1] and MSE is the mean squared error between them.

Figure 92: Reconstruction of Motion JPEG : Reconstruction was performed on a frame–by–frame basis using MATLAB^{TM}. The numbers indicate the frame number.

The fully digital JPEG compression system was implemented in an FPGA using VHDL for power consumption comparison with our system using MATIA. The results are summarized in Table 4. Total power consumption of the fully digital implementation was estimated at 183mW by the FPGA power estimate worksheet. We could save 146mW using the same estimation method by removing the DCT computation part, which is processed in MATIA, from the system. Note that we used a generic DCT module which is not designed for a low power solution. However, considering even low power DCT modules consume about 30∼50mW [94, 85], our system still can achieve significant amount of power improvement.

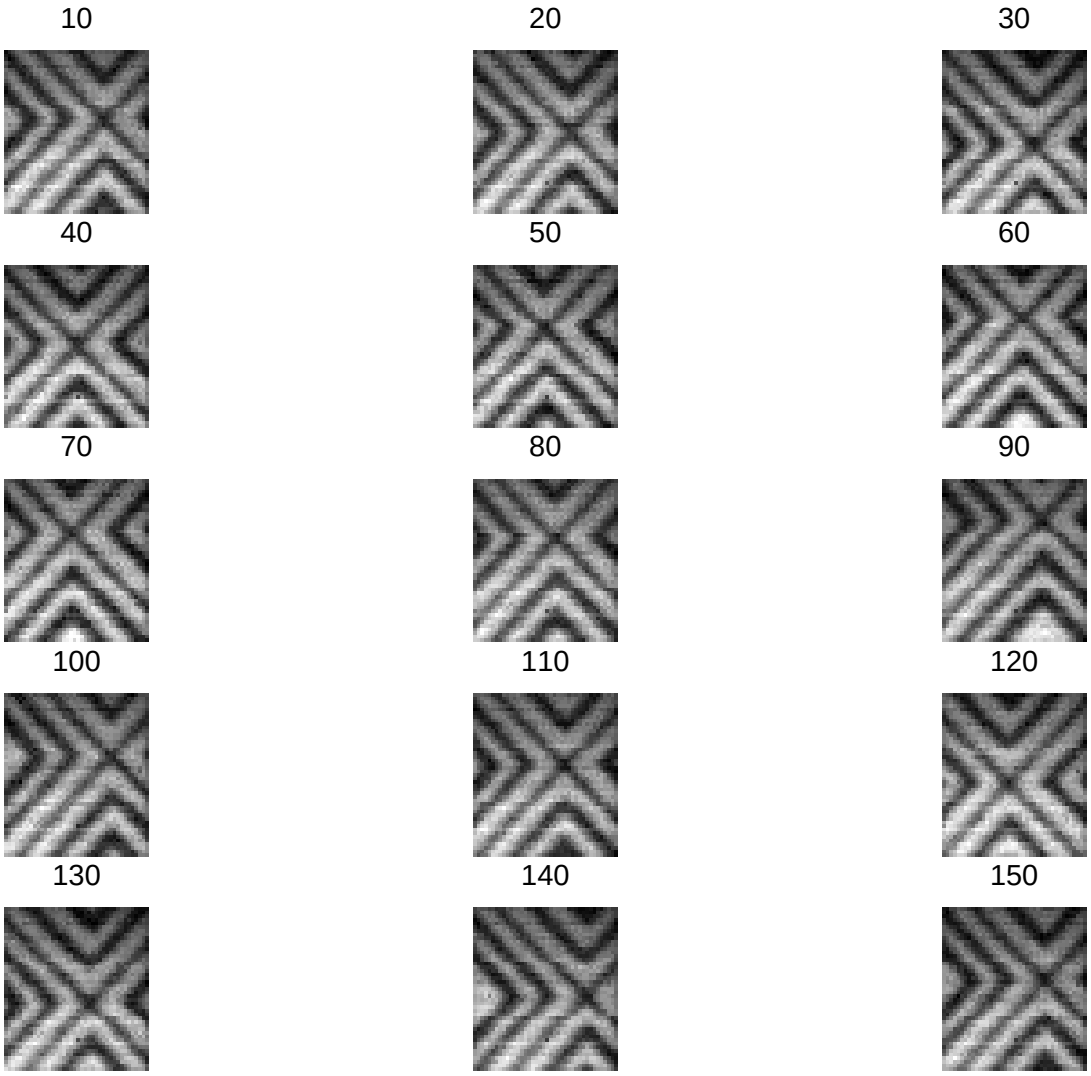


Figure 93: Motion JPEG using Walsh-Hadamard Transform: Motion JPEG was performed on moving images using Walsh-Hadamard Transform. Reconstruction was performed on a frame–by–frame basis using MATLAB^{TM}. The numbers indicate the frame number.

Video compression can be achieved by compressing the frames of the digital video sequence individually. This approach can be used with JPEG compression algorithm and is called motion JPEG. We have used MATIA to implement motion JPEG using DCT and Walsh–Hadamard transforms as the block transforms. Figure 91 shows the output of 16x16 block DCT performed at every frame, while Fig. 92 shows the reconstructed frames. Reconstruction was performed using MATLAB^{TM}. Similarly, Fig. 93 shows the reconstructed frames after on–chip 16x16 block Walsh-Hadamard transform was performed for

Table 5: Summary of MATIA characteristics

Technology	0.5μ N-well CMOS
Array size	104×128
Pixel size	13.5 μm×13.5μm
Fill factor	46%
Kernel size	2×2 to 16×16
Programming error	< 0.2% for 2.5 decades
Programming mechanisms	Hot electron injection and electron tunneling
No. of programmable parameters	6656
Frame rate	25 fps
Dark current	14.9nA/cm^2
Weber ratio	0.1211
Transistor count	74K
Frequency response (pixel)	DC–100KHz
Power consumption($V_{DD} = 3.3V$)	80μW/frame

every frame. Since DCT gives better energy compaction on a block–by–block basis, when compared to Walsh–Hadamard transform, JPEG using DCT would give higher compression for the same image. Although motion JPEG, which only exploits spatial redundancy, reduces the bit rate significantly higher compression rates can be obtained by exploiting spatio–temporal redundancy inherent in video sequences.

The general characteristics of the MATIA chip are summarized in Table 5. Figure 94 shows the die micrograph of our 128x128 MATIA. Much larger arrays are possible without much impact on the performance of the scanning and processor unit. In a larger array most of the additional area will be used in the photo array as overhead required for the scanning and processing will be similar to that in this chip.

5.8 Conclusion

MATIA is enabled by programmable floating–gate circuits built in standard CMOS (single or double-poly) processes. The floating-gate circuits allow for arbitrary pattern generation as well as analog matrix-vector multiplication of images. This architecture is capable of performing different matrix operations on an image. The pixel used in this architecture performs matrix multiplication while maintaining a high fill-factor (46%), comparable to

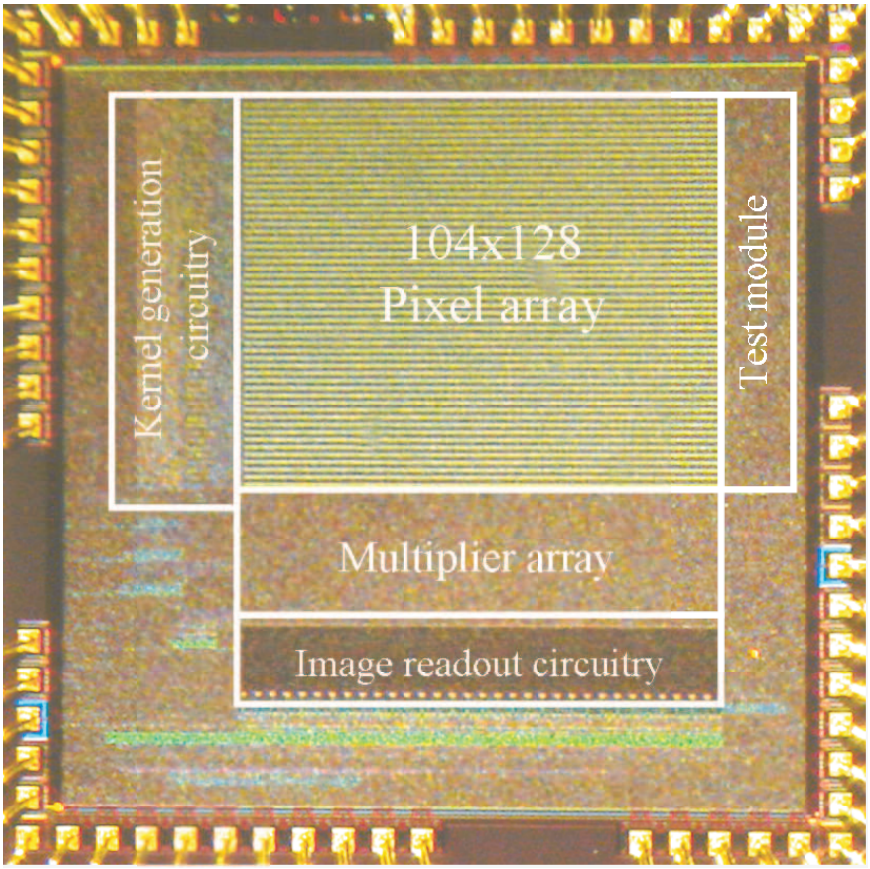


Figure 94: Die Micrograph: This 128×128 MATIA chip was designed in 0.5μ N-well CMOS technology. The chip has a photosensitive area of $2.99mm^2$

active pixel sensors. Floating gates are used to store the arbitrary matrix coefficients on–chip. The chip operates in the subthreshold domain and thus has low power consumption (80μW/frame). Since this architecture is fully programmable various transforms can implemented using the same architecture. We have illustrated the use to MATIA for JPEG and motion JPEG systems. The resulting data-flow architecture directly allows computation of spatial transforms, motion computations, and stereo computations, in a straightforward on-chip or multi-chip architecture.

CHAPTER VI

APPLICATIONS AND IMPACT

6.1 Impact of this work

Previously there were two schools of thought in imagers: APS and neuromorphic imagers. Groups working with APS imagers have traditionally concentrated on large and fast imagers used only for reading images. Their main concern was accuracy and speed for the imagers. Neuromorphic imagers on the other hand have concentrated on emulating functionality of biological systems. As as result these imagers have smaller fill factor but perform complex computation at the pixel–level. MATIA is the first architecture that performs high level computation while maintaining a high fill factor. This imager is also fully programmable and hence can be used in various applications as discussed above. The impact of this work can be summarized as below:

1. *Adaptive programming large floating gate arrays*: I, along with Mr. Guillermo Serrano, have developed an adaptive programming algorithm for accurately programming large arrays of analog computational memory elements within 0.2% of accuracy for 3.5 decades of currents. The average number of pulses required are 7-8 (20μs each). This algorithm uses hot–electron injection for accurate programming and Fowler-Nordheim tunneling for global erase. This methodology has been tested for programming large floating–gate arrays in 0.25μm and 0.5μm N-well CMOS processes. This is the first time that an algorithm can effectively program both in the sub–threshold and the above threshold region using only one prior characterization. This is also the first time that large floating–gate arrays have been programmed with this accuracy and speed. The accuracy, right now, is limited by the resolution of the onchip I-Vs used for measuring the currents and also by the accuracy of the DACs used for pulsing the drain.

The proposed method is computationally intense. It requires the storage of characterization parameters for a family of V_{DS}. A linear regression also has to be performed before the exact V_{DS} can be calculated. We have also proposed and proved a simplified model for this algorithm that has the same accuracy. Using this method only six parameters have to be measured and stored. Since this method is a direct calculation, the computational complexity is reduced, and no regression has to be performed.

These algorithms have been extensively used by several members of the lab. It has been tested over processes and with different sized floating gates in various systems, like DACs, FPAAs, audio systems and imagers.

I have designed, fabricated and characterized different capacitors (double poly and MOS capacitors) that can be used with floating gates. Faster processes (0.25μm TSMC etc.) do not allow double poly capacitors and this characterization was performed to investigate if MOS capacitors can be used, instead of the usual double poly capacitors, in the floating gate arrays.

2. *Designing, analysis, simulation and testing of single pixel structure*: I have designed, simulated and tested single pixel structures in various processes for functionality. This pixel has been used in larger systems for image transforms. This pixel can be used for either reading an image or performing multiplication in the subthreshold region. Since the operation is in the subthreshold region the overall power consumption is less. I have also analyzed the pixel for noise and SNR, threshold mismatches, harmonic distortion, gain errors, and dark currents. I, along with Mr. Ryan Robucci, have measured the above mentioned parameters.

3. *Designing, analysis, simulation and testing of current–mode four quadrant vector matric multipliers*: I, along with Mr. Ravi Chawla and Mr. Venkatesh Srinivasan, have designed, simulated, tested and analyzed a 128x32 current–mode analog vector–matrix multiplier (VMM). This fully–differential current–mode VMM architecture is fully programmable.

   In order to achieve high power efficiency and low power operation, a sub-threshold

implementation is ideal. A voltage mode implementation operating in sub-threshold will have limited linearity due to the exponential I-V relationship of the transistor operating in saturation. This limitation in linearity can be alleviated to a certain extent by using methods like source degeneration, which degrades the frequency response for a particular power. A current-mode implementation can be used to overcome some of these limitations.

The architecture is suitable for low power applications and has bandwidth-to-frequency ratio of 531nW/MHz per differential multiplier. For a bandwidth of less than 10MHz, this architecture is capable of performing 1 million MAC/0.9μW as compared to a commercially available DSP (TMS32005x series), which gives 1 million MAC/0.25mW. The VMM chip gives a linearity of over 2 decades with a worst case error of $\pm$2.5%. The area of the chip is 0.83mm^2. The IC prototype was fabricated in a 0.5μm CMOS MOSIS process. We have demonstrated block matrix transforms using this architecture. The VMM chip can be used for applications like audio and video processing.

4. *A working system for block matrix transforms*: I have designed, simulated and tested a low power image processor. This imager is capable of performing programmable matrix operations on an image. The imager architecture is both modular and programmable. The pixel used in this architecture performs matrix multiplication while maintaining a high fill-factor (46%), comparable to active pixel sensors. Floating gates are used to store the arbitrary matrix coefficients on–chip. The chip operates in the subthreshold domain and thus has low power consumption (80μW/frame). I have tested the various sub-blocks of the imager system for functionality. The various decoders and other peripheral circuits, used for automatic programming and image read–out have been tested individually. These system chips were designed for differential current outputs. Peripheral circuits for automatic programming of large arrays using an external programming board were also included. These system chips were designed so that each sub-block could be tested individually.

   I have successfully programmed and acquired images and system data from 14x14,

16x16, 64x72 and 128x128 imagers that were fabricated using 0.5μm AMI technology. I have also gathered data which prove that the system architecture can be used for different types of on–chip filtering and transforms.

5. *A low power reconfigurable JPEG compressor*: I have implemented a low pow JPEG compressor using MATIA. By partitioning the algorithm into analog and digital parts, significant amount of power improvement can be achieved compared to the conventional digital implementation. Peak signal–to–noise ratio (PSNR) of compressed images with different bit per pixel (bpp) are measured to verify the operation and the performance of the system, and power improvement was verified using our FPGA only implementation.

   The fully digital JPEG compression system was implemented in an FPGA using VHDL for power consumption comparison with our system using MATIA. Total power consumption of the fully digital implementation was estimated at 183mW by the FPGA power estimate worksheet. We could save 146mW using the same estimation method by removing the DCT computation part, which is processed in MATIA, from the system. Note that we used a generic DCT module which is not designed for a low power solution. However, considering even low power DCT modules consume about 30$\sim$50mW, our system still can achieve significant amount of power improvement. I have demonstrated the use of MATIA for motion JPEG using DCT and Walsh–Hadamard block transforms.

   I have designed and populated a PCB that can be used for faster image capture and programming of floating gates. The PCB is to be controlled by a FPGA. The layout was done by Mr. Faik Baskaya. The VHDL code for testing the system imager was written by Mr. Jungwon Lee.

6. *An optical bench for universal imager testing*:

   I have designed a versatile test setup for testing our imagers. This setup can be used to test any imager that have been designed for still or video applications. The test setup is interfaced with the computer such that arbitrary images can be focussed onto

the chip being tested. It is designed so that arbitrary images or video can be focussed onto the photodiode array area of any imager chip. The setup consists of a liquid crystal display (LCD) which is connected to the computer being used. Any image that is displayed on the LCD is focussed onto the chip using complex lenses. I have assembled and tested this testing bench.

The work finished has been published in seven conferences [7, 8, 19, 57, 58, 59, 95], five journal papers [9, 10, 11, 20, 53] and one patent [34]

6.2 Applications for MATIA

The MATIA architecture can be used for various other applications other than as transform imagers or as image encoders. This imager can be used for any applications that need block transforms of images for operation. It can also be used just as an imager as it can be programmed to read an image using the same architecture.

6.2.1 Depth from Stereo

Figure 95 shows a block diagram using these transform imagers to compute depth from two imagers (stereo image processing). Since the two imagers have pixels aligned along one axis, computing depth from stereo only requires computing offsets on a row–by–row basis. In our architecture, the pixel offset function for each row can be calculated using either a correlation–based technique or, with slight modification, an absolute difference squared approach. The pixel offset function is calculated instantaneously for each row as it is presented to the stereo correlation circuit from the two transform imagers. One advantage of using the transform imagers is that calibration can be performed on the individual imagers making the stereo processing possible in easily manufactured systems. It should also be noted that the correlation–based stereo pixel offset calculation is similar to Maholwald's one-dimensional stereo imager design [82] based on models of perception [104]. One could also imagine similar algorithms to compute three-dimensional motion enabled by two or more imagers.

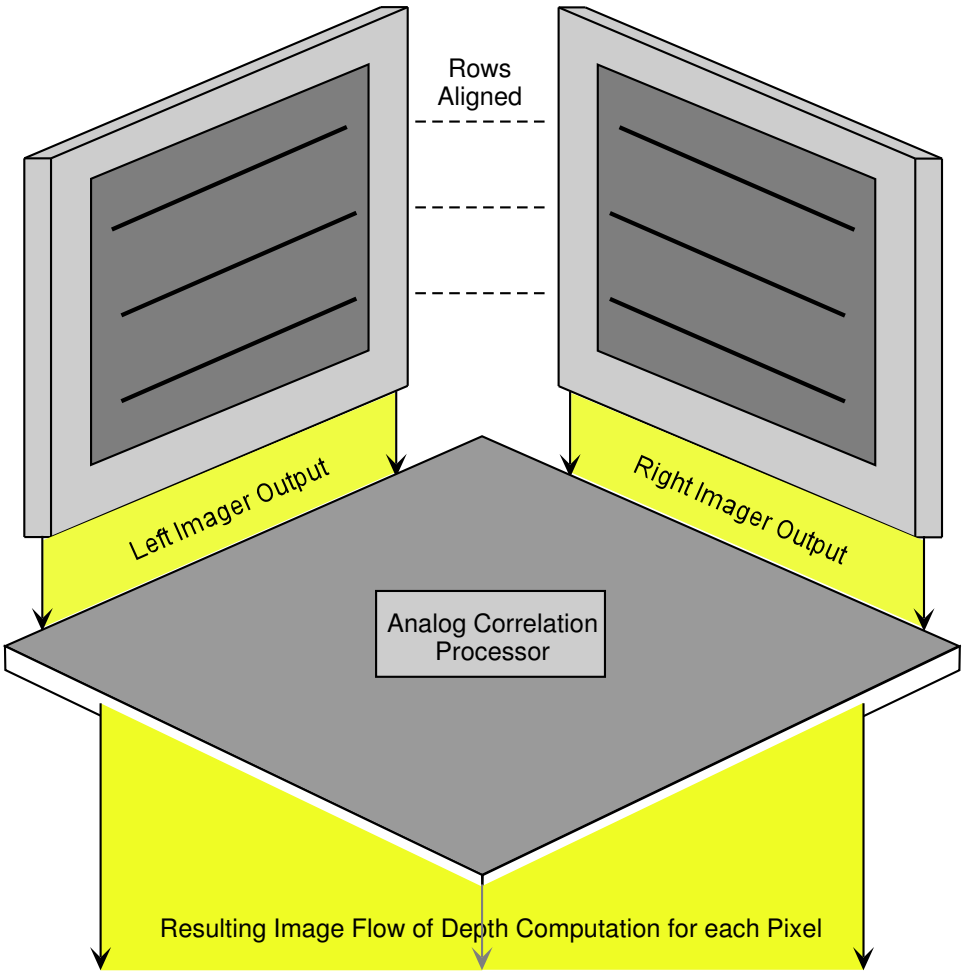


Figure 95: Using two transform imagers to compute stereo computation: Because each row is aligned with another row on the second imager along the same axis, finding depth for each pixel is a row by row operation, which maps directly into the transform imager's output. The inputs to the stereo processing could be outputs from two imagers, or any symmetric processing starting from two imagers. The resulting computation is similar to Maholwald's one-dimensional stereo imager design, although a variety of algorithms can be computed using this architecture.

6.2.2 Temporal filtering

One interesting question with this flow model is how to perform temporal filtering. We can either build the filters directly into the pixel, which would result in much larger pixels and greatly increase the system cost for a given resolution, or we can store a delayed version of the transformed image. This approach requires a temporary storage array for currents or voltages for each delay thus limiting the number of temporal delays that can be built in practice (Fig. 46).

Applications of temporal filtering include subtraction of constant background images,

temporal differencing, motion estimation, and, by using an array of floating-gate elements instead of the sample–and–hold elements, *fixed* images such as offset errors from dark currents may be subtracted out. In general, however, temporal filters should be used sparingly or after spatial compression due to the number of required sample–and–hold elements.

6.2.3 Universal Matrix Image Transforms

MATIA can be used to test any arbitrary transforms on silicon since it is fully programmable. It compute arbitrary separable 2-D linear operations. These operations are expressed as two matrix multiplications on the image

$$\mathbf{Y} = \mathbf{A}^T\mathbf{P}\mathbf{B} \tag{57}$$

where $\mathbf{P}$ is the image array of pixels, $\mathbf{Y}$ is the computed output image array, and $\mathbf{A}$ and $\mathbf{B}$ are the transform matrices corresponding respectively to the transform on the image plane by the basis functions and the the floating-gate enabled multiplication after the image plane. Furthermore, if the input waveforms are continuous, then the result is a continuous waveform, resulting in added computational options. For example, the choice of output signal sampling will result in different discrete–time inspired computations with an identical setup.

6.2.4 Preprocessing for Optical Flow

Figure 96 shows the block diagram using these imagers and additional matrix computations as a front-end processor to compute optical flow. One must design separate spatial (for x and y directions) and temporal differencing blocks. The spatial differencing is performed using a multi-resolution differencing matrix so that subsequent processing can identify smooth large–object motion as well as detail motion. The temporal differencing block requires a set of current sample–and–hold elements built into an array that could be used for computing temporal derivatives of this image. These outputs can then be used by digital processing to compute global displacement estimation, target location, and other higher–level information with reduced complexity.

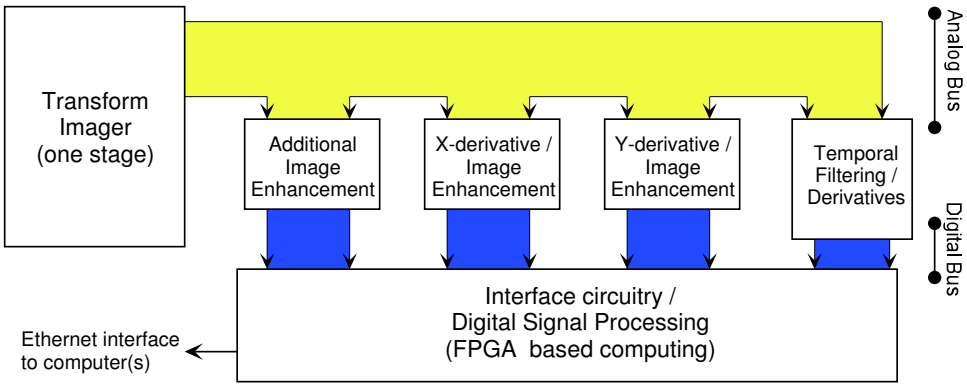


Figure 96: Block diagram for a motion (optical flow) computation system: This system computes the two spatial derivatives (x and y), a temporal derivative, and a filtered version of the original image. These derivatives are required for any optical flow system. We can make fairly smooth derivative operations in the spatial domain by combining a smoothing filter with the derivative kernel over a moderate window size (16x16 or greater). The time derivative is computed by subtracting two or more successive frames, which require a sample and hold for each image frame.

6.2.5 DCT based classification

Texture as a primitive visual cue has been studied for a long time. Various techniques have been developed for texture segmentation, texture classification and texture synthesis. In general, neighboring pixels within an image tend to be highly correlated. As such, it is desired to use an invertible transform to concentrate randomness into fewer, decorrelated parameters. The Discrete Cosine Transform (DCT) has been shown to be near optimal for a large class of images in energy concentration and decorrelating. The idea is to use a few of the most "informative" coefficients to represent the image space, and classify each image based on these coefficients.

Most of these algorithms have been looked at from a software point of view, but using the MATIA these algorithm can be tested on silicon. Since this architecture is current–mode other current mode post–processing can be performed on images.

6.2.6 Super–resolution techniques for high resolution images

The idea of super-resolution, combining images by combining pieces from an image sequence into a single image with higher resolution than any of the individual images, has been around for years. Every image in a sequence provides some additional information, provided they are not noise–free, focussed, and Nyquist sampled.

The MATIA chip is capable of fast image readout. The factors that slow down the architecture are subthreshold operation and the ADCs that are used for image capture. Standard current boosting and cascoding techniques can be used to circumvent the first limitation. The imager can be operating in the subthreshold region but the current outputs can be boosted for faster operation of the subsequent stages. The ADC that is being used right now is the integrating type. This is a slow but accurate ADC. Faster ADCs, like pipelined, successive approximation or flash ADCs can be implemented, in either voltage or current mode for faster readout at the cost of higher power consumption. These fast ADC need not be column parallel and the outputs can be pipelined too for faster image readout. The MATIA can be used as a front end in a system implementing super–resolution images. The other operations required for the whole system can be implemented either using DSP techniques or traditional and neuromorphic analog techniques.

APPENDIX A

OPTICAL TEST BENCH

I have designed a versatile test setup for testing our imagers. This setup can be used to test any imager that have been designed for still or video applications. The test setup is interfaced with the computer such that arbitrary images or video can be focussed onto the chip being tested.

Figure 97 shows the the block diagram of the overall setup. The test bench has been designed such that everything is controlled using a main PC or laptop. The setup has two monitors. One of the video outputs go to a video splitter and then to the second monitor and the LCD display. This way, anything that is displayed on the second monitor also comes on the LCD. Hence this setup can be used for testing either still imagers or video imagers, as these can be effectively focussed onto to the test chip. The setup also consists of an FPGA board and the optical test bench. The VHDL modules for controlling the test board and also for image capture run on the FPGA. This FPGA is controlled though MATLAB^{TM} running on the main PC/laptop.

Figure 98 shows the schematic of the optical test bench. A DC regulated light source is used as a back light for the LCD screen, so that AC interference is minimized. The LCD is interfaced with a computer through a video splitter. Since the LCD (1024×768) needs polarized light for operation two polarizers have been used. The image is then focussed onto the light sensitive area of the imager chip through a complex lens system. The chip is mounted onto a XYZ translator so that the image can be focussed onto the chip accurately. All of the components used are mounted on a optical slide for easier movement along one plane. Figure 99(a) and (b) show the front view and the side view of the optical test bench. It also shows how the PCB and the test chip are mounted onto the optical test bench. Table 6 lists the parts required to assemble the optical test bench.

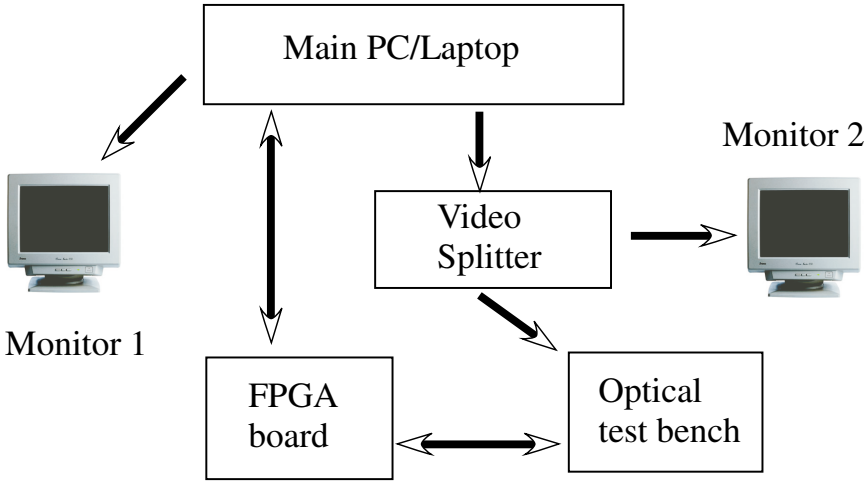


Figure 97: Imager test setup Schematic of the universal imager test setup

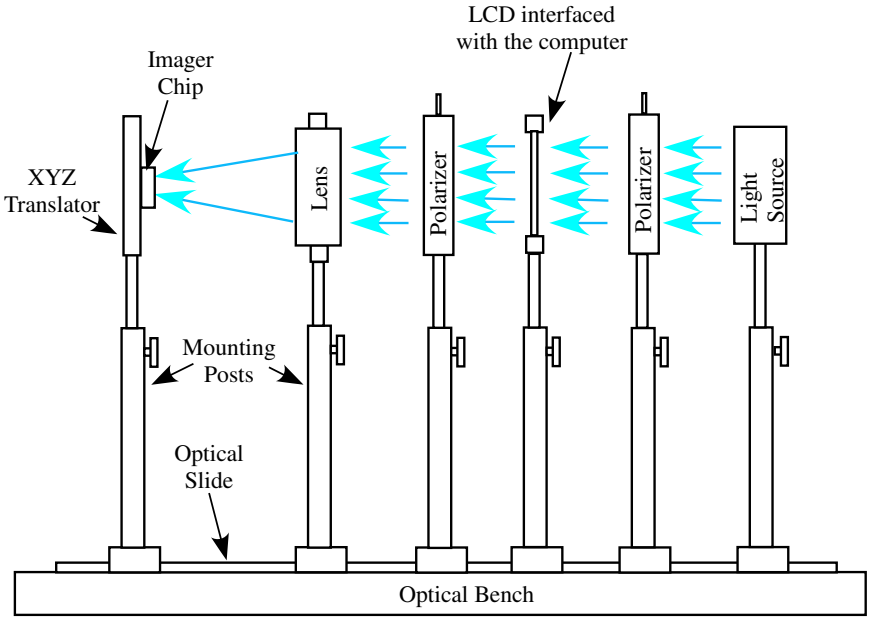
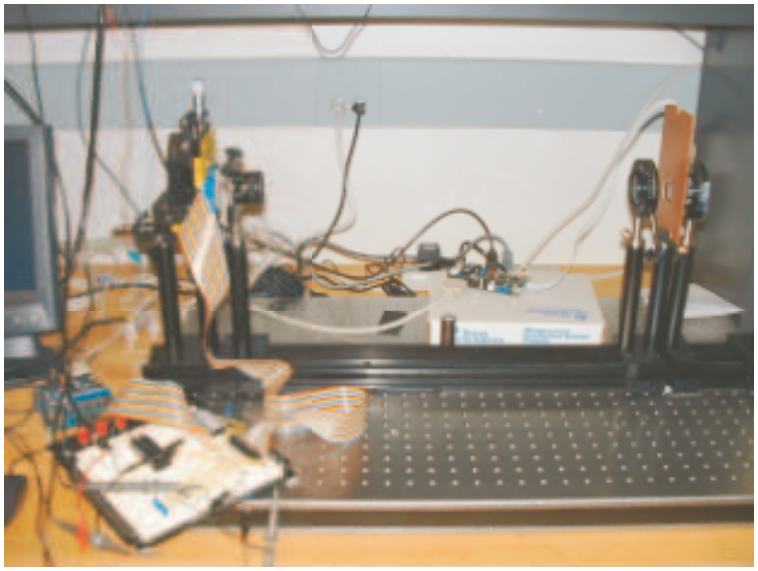
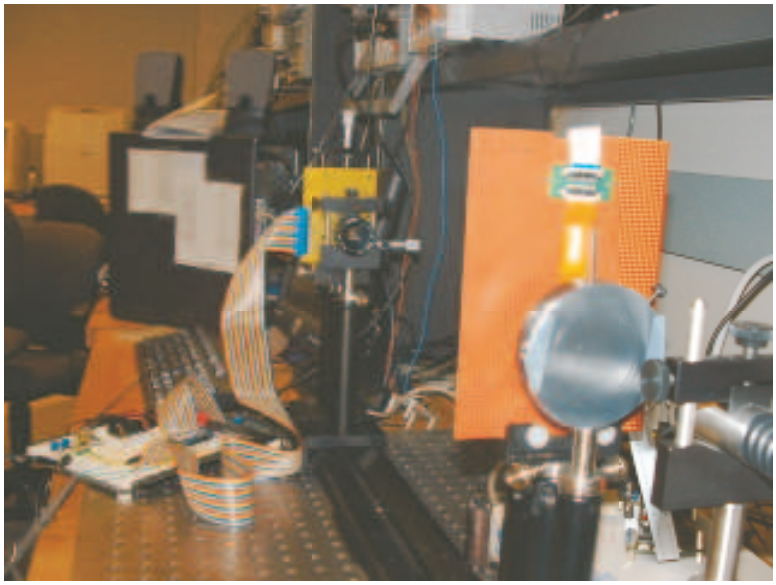


Figure 98: Optical test setup Schematic of the universal imager test setup. It can be used for both still–image and video chip testing.



(a)



(b)

Figure 99: Optical test setup Different views of the optical test setup

Table 6: Parts for the universal test setup

Optical rail, Dovetail, 36”	Carrier: 2.5”,1.5”, 0.5”
Wratten neutral density filter	Std. Post holders, 6” (10) (0.1, 0.3, 0.4, 0.6,0.9, 2) O.D.
Mounting post, 6” (10)	Composite breadboard table 23”x35”
Post colar lock ring (10)	Optical cell assembly (3)
Medium 2.62” square translation stage	X–Y Metric stage three axis (X–Y–Z)
Mounted polarizing filters (2)	Rotating polarizer holder (2)
Lens kit	Mounted std. Iris diaphragm
Allen wrench	Allen head set screw
Flat head Phillips screw	Heat absorbing glass 50mm x 50mm
Fixed filter mount	Beam splitter (2)
Rectangular optical mount (2)	Prism holder (2)
Bar–type lens mount 40mm (2)	PCX lens 40mm
1/4–20 English socket head cap screw	Angle mount, 3”
Video camera, Monochrome high resolution	Fixed focal length lens, Manual Iris, 25mm
Fixed focal length C–mount spacer kit	DC regulated fiber optic illuminator PL-900
Fiber optic light guide adapter SX–10	2 port VGA video splitter
Video card	Video cables
GPIB card	LCD display (1024×768)
ball driver (3/16)	TV 13” monitor

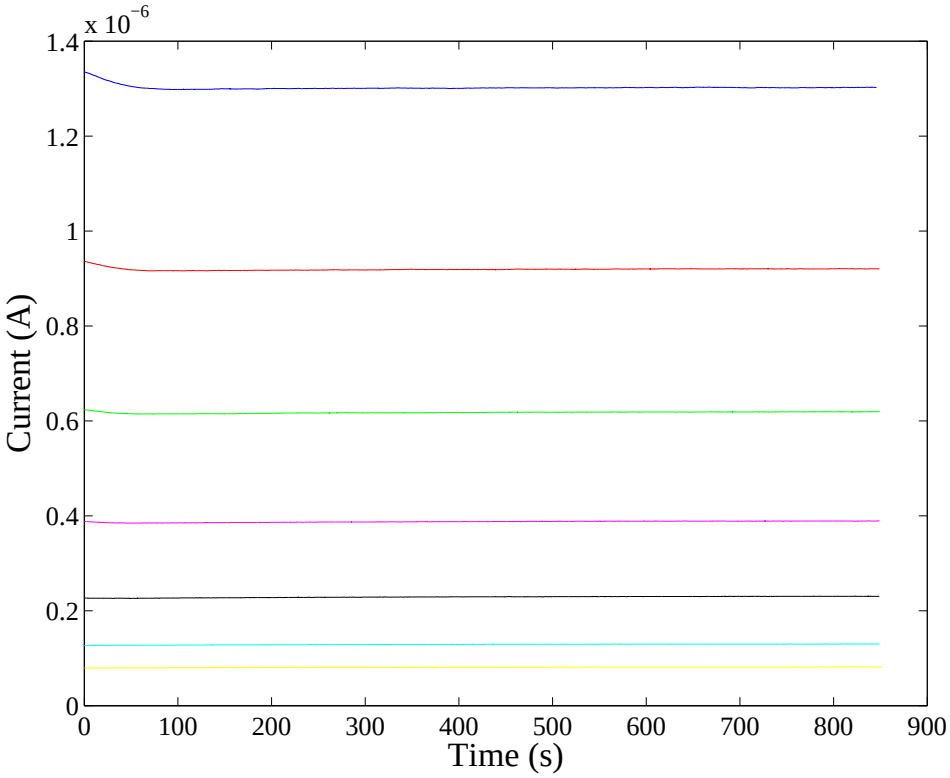
APPENDIX B

FLOATING GATE RAMPUP/RAMPDOWN TRANSIENTS

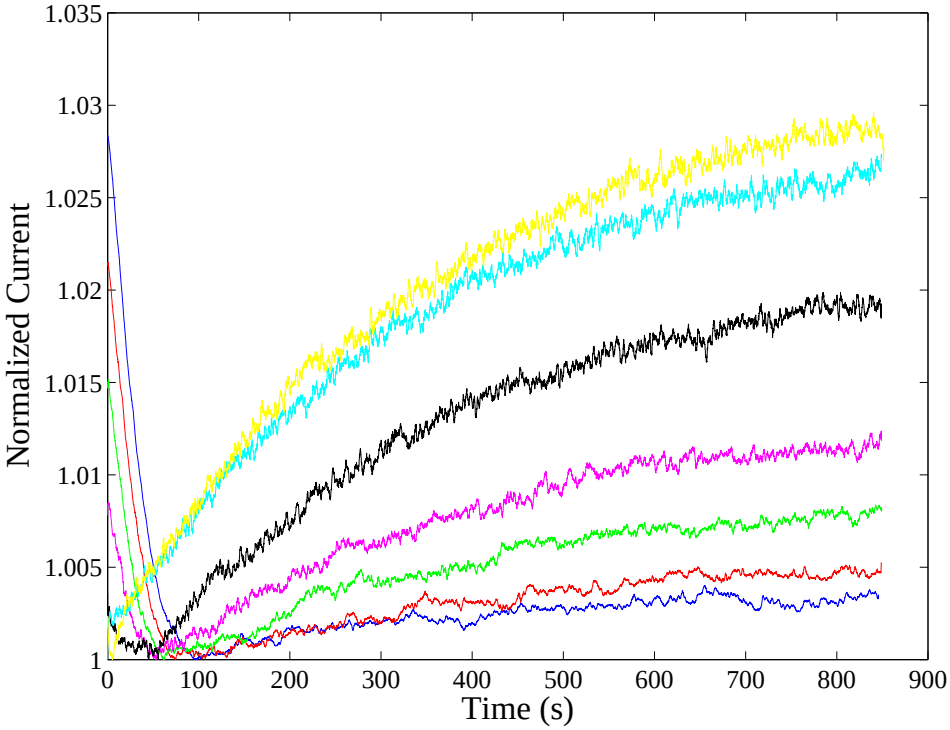
The floating gates are programmed using FN tunneling and Hot–electron injection processes. For injection the chip has to be *ramped up* and *ramped down*, as explained in Chapter 2. When these are performed some current transients are observed. Measured currents seem to change over a period of minutes after injecting.

The transients for different initial currents are plotted in Fig. 100. It can be observed that lower ramp up currents induce longer transients and thus takes a longer time to settle to a final value. Similarly the transients can be observed during the ramp down phase too, as shown in Fig. 101. Again it is observed that the lower currents induce larger settling times. The transients are also observed for changing supply voltages Ramp down transients for various supply voltages (V_{DD}) as shown in Fig. 102.

These transients have been observed in almost all chips fabricated in 0.25μ and 0.5μ N-well CMOS processes. Much thorough study needs to be conducted to understand these effects.

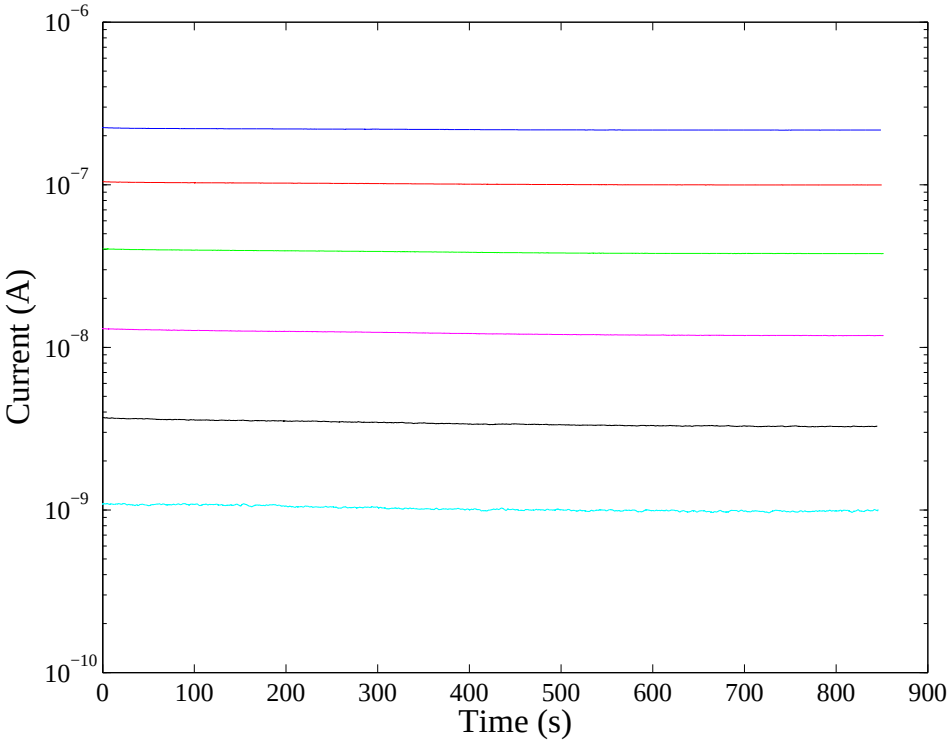


(a)

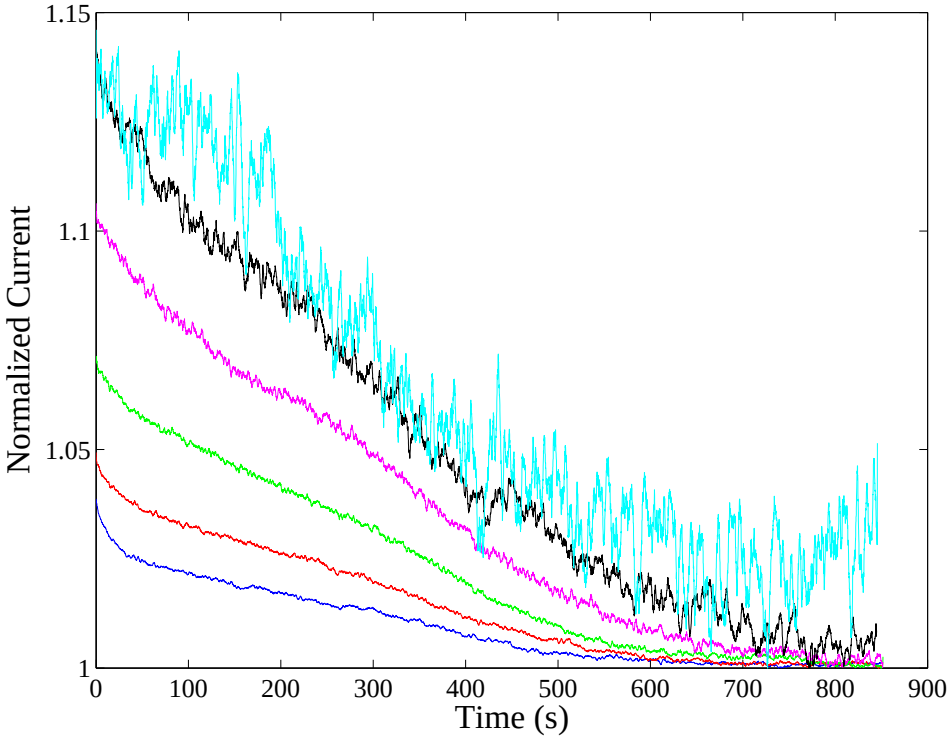


(b)

Figure 100: Ramp up transients for various initial currents: (a) Initial measured ramp up currents, (b) Normalized currents for comparing transients.

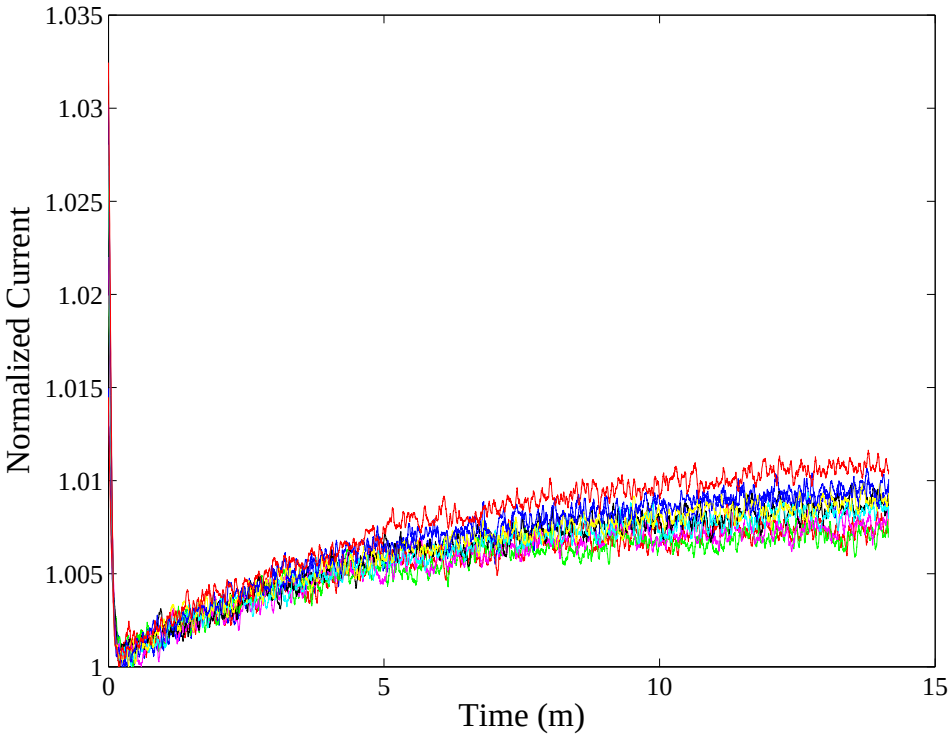


(a)

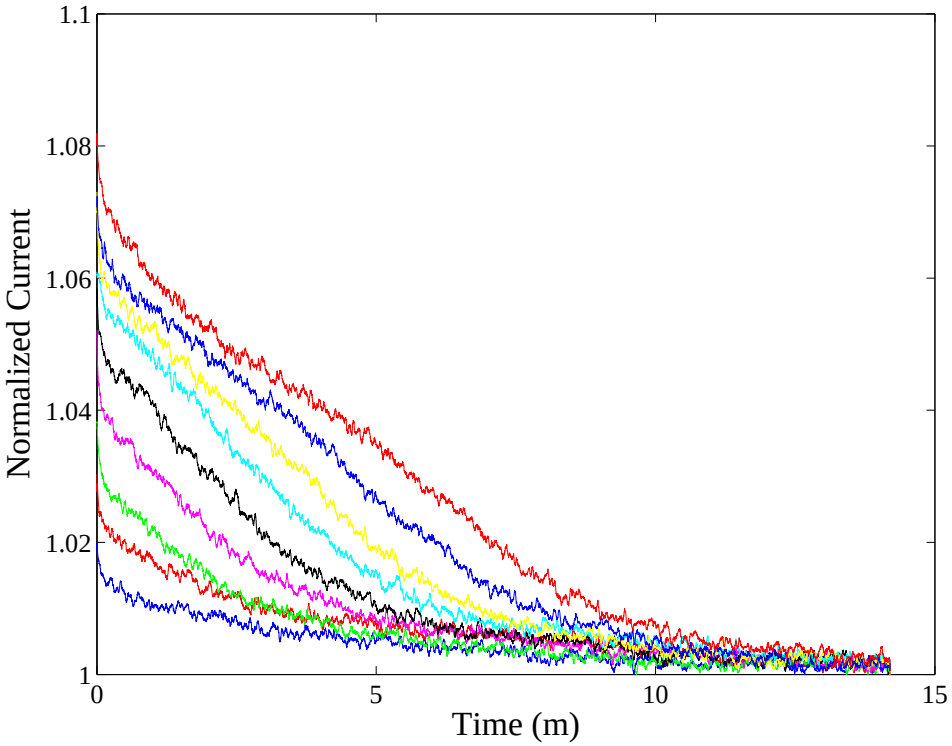


(b)

Figure 101: Ramp down transients for various initial currents: (a) Initial measured ramp down currents, (b) Normalized currents for comparing transients.



(a)



(b)

Figure 102: Ramp down transients for various supply voltages (V_{DD}): (a) During ramp up, (b) During ramp down.

APPENDIX C

PRINTED CIRCUIT BOARD

Two printed circuit boards were designed for testing, programming, and fast image capture from the imager chips. Figure 103 and Fig. 104 show the schematic of the second board. The features of the board are as follows:

- The analog biases (drain voltage, gate voltage, tunneling voltage, and power supply) required by the imager chip are provided by octal-DACs which are controlled by a FPGA. There are additional circuitry for generating the tunnel voltage (15 V) and all the analog voltages are buffered before they are presented to the imager chip.
- All the DACs are controlled through FPGA.
- The board board also has 14-bit 10 MS/s ADCs for image capture.
- The digital signals to be used during the rampup phase go through level shifters, since a high power supply (6V 6.5V) is required for programming.
- The data was transferred from the FPGA to the computer through an ethernet connection.
- The VHDL modules for controlling the different modules in the board were controlled using a C core. This was in turn controlled by MATLAB^{TM}. The test setup was configured such that it can be fully controlled through MATLAB^{TM}.

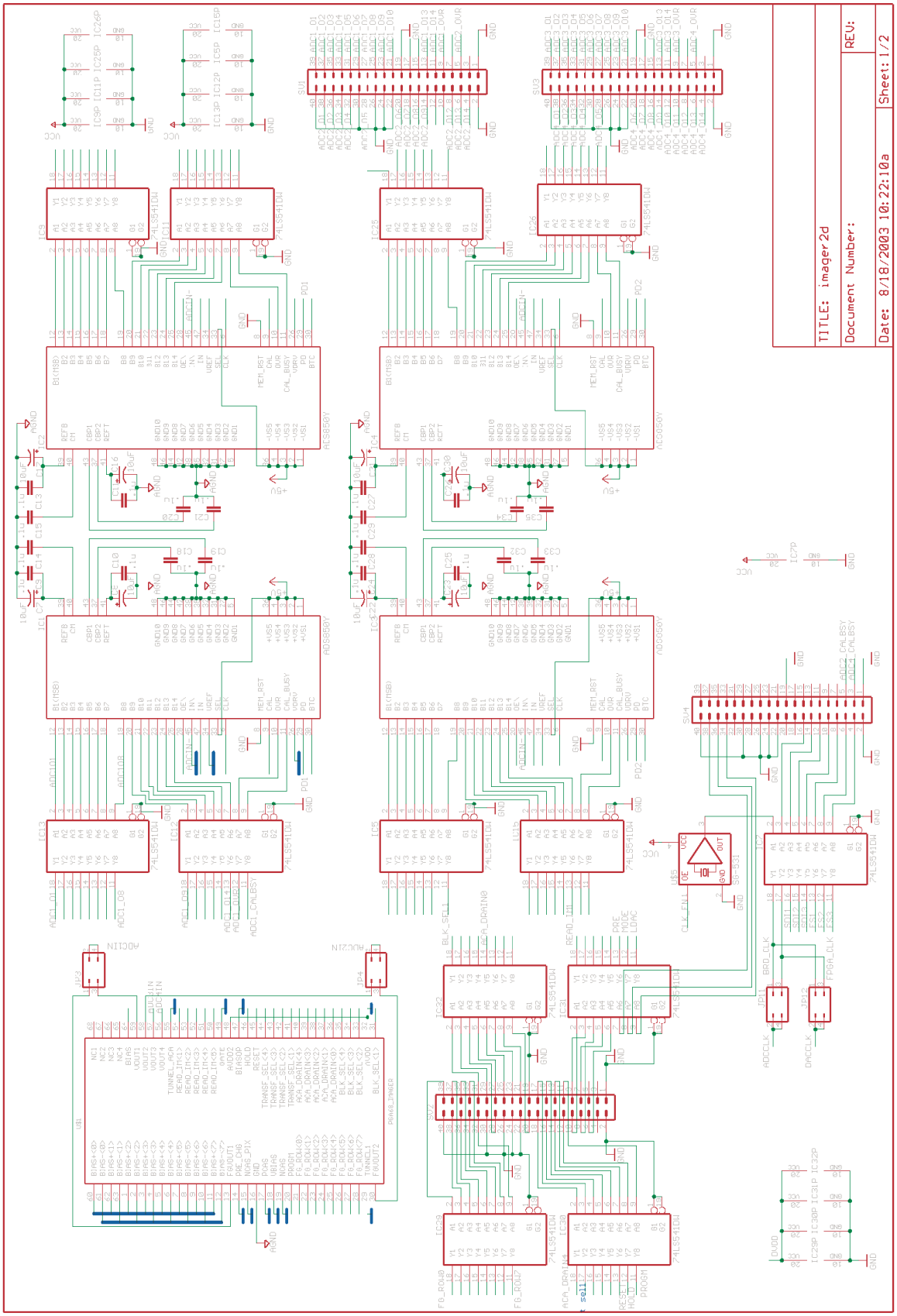


Figure 103: Schematic of imager board - I: Schematic of the imager board. It was designed using EAGLE.

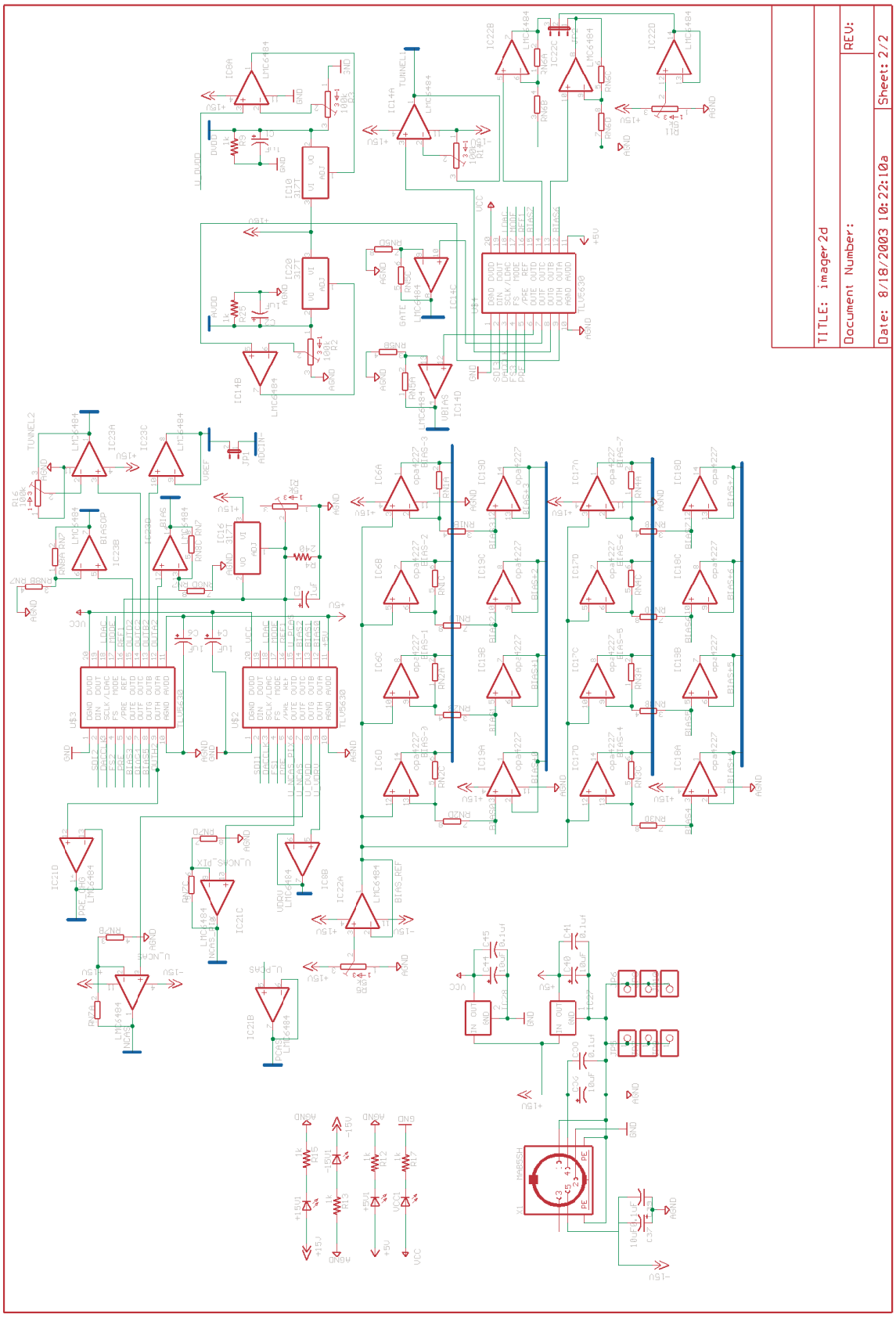
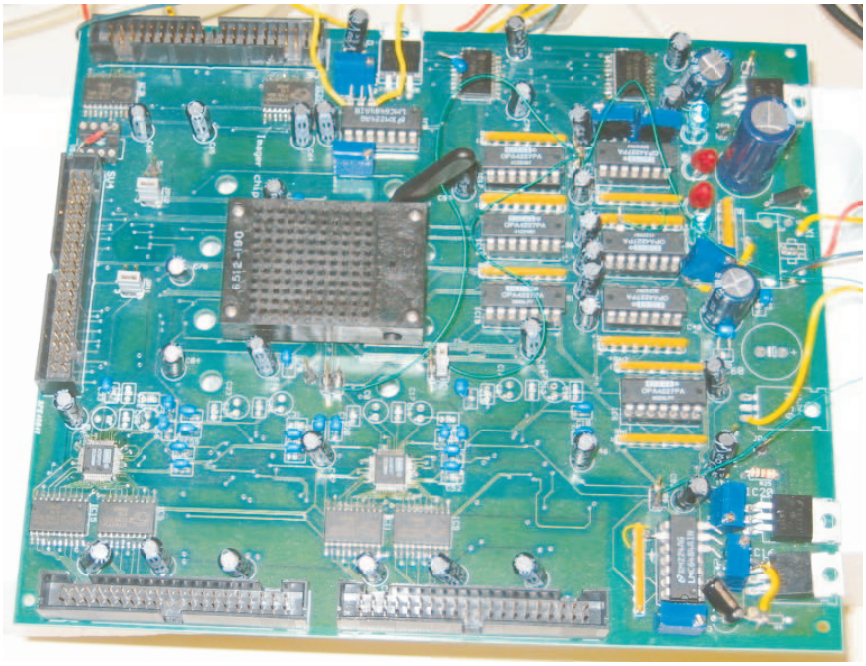
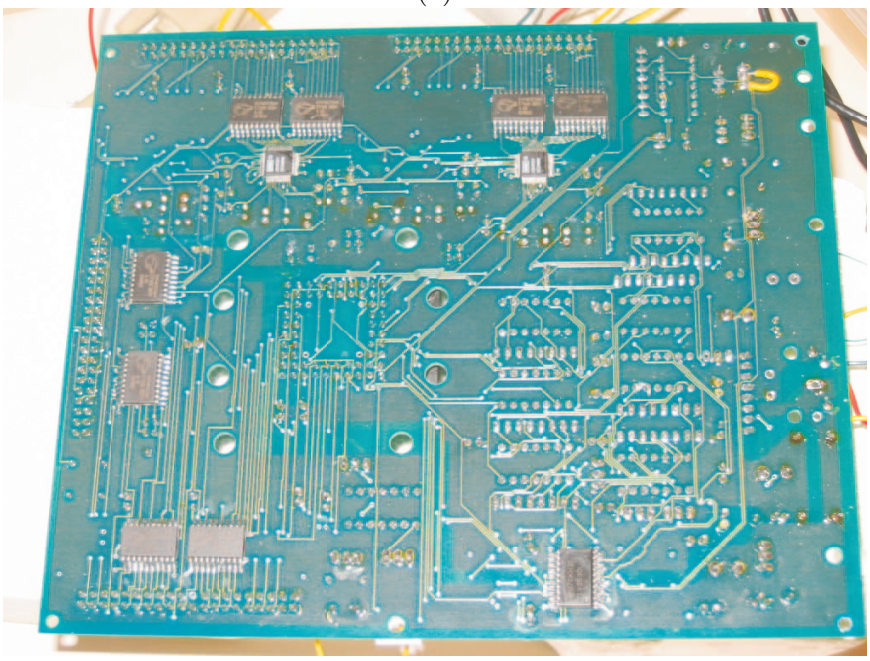


Figure 104: Schematic of imager board - II: Schematic of the imager board. It was designed using EAGLE.



(a)



(b)

Figure 105: Pictures of PCB: (a) Top, and (b) bottom view of the PCB.

APPENDIX D

DESCRIPTION OF CHIPS FABRICATED

The following chips have been fabricated:

Pixel characterization chips	
Chip no.	**Description**
T16W_CR	Pixel Characterization chip (AMI 05)(Hasler, Smith)
Imager system chips	
Chip no.	**Description**
T21R_EF	16x16 system chip using poly capacitors for floating gate capacitors (AMI 05)
T21R_DU	16x16 system chip using MOS capacitors for floating gate capacitors (AMI 05)
T21S_BC	128x128 system chip using poly capacitors for floating gate capacitors (AMI 05)
T26Y_BG	Modified 128x128 system chip using poly capacitors for floating gate capacitors (AMI 05)
T2AK_AU	16x16 system chip using poly capacitors for floating gate capacitors with ACA (AMI 05)
T29U_AP	512x460 system chip (TSMC 0.25)
T29U_PR	16x16 system chip using poly capacitors for floating gate capacitors and having integrating type on-chip A/D (AMI 05)
T29U_PB	48x40 system chip using poly capacitors for floating gate capacitors and having integrating type on-chip A/D (AMI 05)

Peripheral Test structures	
Chip no.	**Description**
T1BD_BQ	14x14 array of current copiers (AMI 05)
T21R_EX	Current mode sigma delta A/D for column parallel readout (AMI 05)
T23R_CC	4x4 system chip using MOS capacitors for floating gate capacitors and having on-chip programming structures(AMI 05)
T23R_BE	Test structures for faster pitch constrained decoders, shifters and counters (AMI 05)

REFERENCES

[1] "IEEE standard definitions and characterization of floating gate semiconductor arrays," June 1998.

[2] Acosta-Serafini, P. M., Masaki, I., and Sodini, C., "Predictive multiple sampling algorithm with overlapping integration intervals for linear wide dynamic range integrating image sensors," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, pp. 33–41, March 2004.

[3] Aizawa, K., Ohno, H., Egi, Y., Hamamoto, T., Hatori, M., Maruyama, H., and Yamazaki, J., "On sensor image compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 543–548, June 1997.

[4] Anderson, D. V. and Hasler, P., "Cooperative analog/digital signal processing," in *World Conference on Systemics, Cybernetics, and Informatics*, (Orlando, FL), July 2001.

[5] Andreou, A. G., "Low power analog VLSI systems for sensory information processing," in *Microsystems technologies for multimedia applications* (Sheu, B., Sanchez-Sinencio, E., and Ismail, M., eds.), pp. 501–522, Los Alamitos, CA: IEEE Press, 1995.

[6] Aslam-Siddiqi, A., Brockherde, W., and Hosticka, B., "A 16 x 16 nonvolatile programmable analog vector-matrix multiplier," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 1502–1509, Oct. 1998.

[7] Bandyopadhyay, A. and Hasler, P., "A fully programmable CMOS block matrix transform imager architecture," in *Proceedings of the Custom Integrated Circuits Conference*, (San Jose, CA), pp. 2233–2236, 2003.

[8] Bandyopadhyay, A., Lee, J., Robucci, R., and Hasler, P., "A 80μW/frame 104x128 cmos imager front end for jpeg compression," *to be submitted*, 2004.

[9] Bandyopadhyay, A., Lee, J., Robucci, R., and Hasler, P., "MATIA: A programmable 80μW/frame CMOS block matrix transform imager architecture," *to be submitted*, 2004.

[10] Bandyopadhyay, A., P.Hasler, and Anderson, D., "A CMOS floating-gate matrix transform imager," *IEEE sensors, in press*.

[11] Bandyopadhyay, A., Serrano, G., and Hasler, P., "Adaptive algorithm using hot–electron injection for programming analog computational memory elements within 0.2% of accuracy over 3.5 decades," *to be submitted*, 2004.

[12] Bell, J. A., Bruce, J. W., Blalock, B. J., and Stubberud, P. A., "CMOS current mode flash analog to digital converter," *MWCAS*, pp. 272–275, 2001.

[13] Blum, R., Wilson, C., Hasler, P., and DeWeerth, S. P., "A CMOS imager with real-time frame differencing and centroid computation," in *Proceedings of the International Symposium on Circuits and Systems*, vol. 3, (Phoenix, AZ), pp. 329–332, May 2002.

[14] Boahen, K., "The retinomorphic approach: pixel-parallel adaptive amplification, filtering, and quantization," *Analog Integrated Circuits and Signal Processing*, vol. 13, pp. 53–68, May-June 1997.

[15] Boahen, K., "A throughput-on-demand address-event transmitter for neuromorphic chips," in *Advanced Research in VLSI*, (Atlanta, GA), pp. 72–86, 1999.

[16] Boahen, K. and Andreou, A., "A contrast-sensitive retina with reciprocal synapses," in *Advances in Neural Information Processing Systems 4* (Moody, J. E., ed.), San Mateo, CA: Morgan Kaufman Publishers, 1991.

[17] Brennan, K. F., *The Physics of Semiconductors*. Cambridge, UK: Cambridge University Press, 1999.

[18] Cauwenberghs, G., Neugebauer, C., and Yariv, A., "An adaptive cmos matrix-vector multiplier for large scale analog hardware neural network applications," *Joint Conference on Neural Networks*, vol. 1, pp. 507–511, July 1991.

[19] Chawla, R., Bandyopadhyay, A., Srinivasan, V., and Hasler, P., "A 531nW/MHz, 128x32 current mode programmable analog vector-matrix multiplier with over 2 decades of linearity," in *Proceedings of the Custom Integrated Circuits Conference*, (Orlando, CA), 2004.

[20] Chawla, R., Bandyopadhyay, A., Srinivasan, V., and Hasler, P., "A 531nW/MHz, 128x32 current mode programmable analog vector-matrix multiplier with over 2 decades of linearity," *to be submitted*, 2004.

[21] Cho, K. and Fossum, A. I. K. E. R., "A 1.5–v 550-μw 176x144 autonomous cmos aps sensor," *IEEE Transactions on Electron Devices*, vol. 50, pp. 96–105, Jan 2003.

[22] Cho, K., Krymski, A., and Fossum, E. R., "A 1.2 V micropower CMOS active pixel image sensor for portable applications," in *International Solid-State Circuits Conference*, (San Francisco), pp. 114–115, Feb. 2000.

[23] Chung, S., Kuo, S., Yih, C., and Chao, T., "Performance and reliability evaluations of p-channel flash memories with different programming schemes," *IEDM Tech. Dig.*, pp. 295–298, 1997.

[24] Cohen, M. and Cauwenberghs, G., "Floating-gate adaptation for focal-plane online nonuniformity correction," *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 83–89, Jan. 2001.

[25] Cohen, M. and Cauwenberghs, G., "Floating-gate adaptation for focal-plane online nonuniformity correction," *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 83–89, Jan. 2001.

[26] Decker, S., McGrath, R. D., Brehmer, K., and Sodini, C. G., "A 256 x 256 CMOS imaging array with wide dynamic range pixels and column–parallel digital output," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 12, 1998.

[27] del Rio-Fernandez, R., Linan-Cembrano, G., Dominguez-Castro, R., and Rodriguez-Vazquez, A., "A mismatch-insensitive high-accuracy high-speed continuous-time current comparator in low voltage cmos," *2nd IEEE-CAS Region 8 Workshop on Analog and Mixed IC Design*, pp. 111–116, 1997.

[28] Delbrück, T., "An electronic photoreceptor sensitive to small changes in intensity," in *Advances in Neural Information Processing Systems 1* (Touretzky, D. S., ed.), pp. 720–727, Morgan Kaufman, 1988.

[29] Delbrück, T., "Silicon retina with correlation-based velocity-tuned pixels," *IEEE Transactions on Neural Networks*, vol. 4, no. 3, pp. 529–541, 1993.

[30] Delbrück, T. and Mead, C. A., "Time-derivative adaptive silicon photoreceptor array," in *Proceedings SPIE, Infrared sensors: Detectors, Electronics, and Signal Processing*, vol. 1541, (San Diego, CA), pp. 92–99, July 1991.

[31] DeWeerth, S., "Analog VLSI circuits for stimulus localization and centroid," *International Journal of Computer Vision*, pp. 191–202, 1992.

[32] Doswald, D. and et. al., "A 30-frames/s megapixel real time CMOS image processor," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 1732–1743, Nov. 2000.

[33] Dudgeon, D. E. and Mersereau, R. M., eds., *Multidimensional Digital Signal Prosessing*. Englewood Cliffs, New Jersey: Prentice-Hall, 1984.

[34] Duggar, J. D., Hall, T. S., Hasler, P., Anderson, D. V., Smith, P. D., Kucic, M. R., and Bandyopadhyay, A., "Floating–gate analog circuit." U.S. Patent Application No. 20030183871, 2003. Patent Pending.

[35] Etienne-Cummings, R., van fer Spiegel, J., and Mueller, P., "A focal plane visual motion measurement sensor," *IEEE Transactions on Circuits and Systems*, vol. 44, pp. 55–66, Jan. 1997.

[36] Etienne-Cummings, R., Kalayjian, Z. K., and Cai, D., "A programmable focal-plane MIMD image processor chip," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 64–73, Jan. 2001.

[37] Fossum, E. R., "CMOS image sensors: electronic camera on a chip," in *International Electron Devices Meeting*, (Washington, D. C.), pp. 17–25, Dec. 1995.

[38] Fossum, E. R., "CMOS image sensors: electronic camera-on-a-chip," *IEEE Transactions on Electron Devices*, vol. 44, pp. 1689–1698, Oct. 1997.

[39] Fossum, E. R., "Digital camera system on a chip," *IEEE Micro*, vol. 18, pp. 8–15, May 1998.

[40] Fowler, R. H. and Nordheim, L., "Electron emission in intense electric fields," *Proceedings of the Royal Society of London*, vol. A119, pp. 173–181, 1928.

[41] Fujimori, I., Wang, C., and Sodini, C., "A 256256 CMOS differential passive pixel imager with fpn reduction techniques," *IEEE Int. Solid-State Circuits Conference*, pp. 106–107, Feb. 2000.

[42] Fujishima, H., Takemoto, Y., Onoye, T., and Shirakawa, I., "An architecture of a matrix-vector multiplier dedicated to video decoding and three-dimensional graphics," *IEEE Transactions on Circuits and Systems II*, vol. 9, pp. 306–314, Mar. 1999.

[43] Funatsu, E., Hara, K., Toyoda, T., Miyake, Y., Ohta, J., and Kyuma, K., "An artificial retina chip made of a 128x128 pn–pn variable sensitivity photodetector array," *IEEE Photonics Technology Letters*, vol. 7, pp. 188–190, Feb. 1995.

[44] Funatsu, E., Nitta, Y., Miyake, Y., Toyoda, T., Ohta, J., and Kyuma, K., "An artificial retina chip with current-mode focal plane image processing functions," *IEEE Transactions on Circuits and Systems*, vol. 44, no. 10, pp. 1777–1782, 1997.

[45] Gealow, J. C., Herrmann, F. P., Hsu, L. T., and Sodini, C. G., "System design for pixel-parellel image processing," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 4, Jan. 1996.

[46] Geelen, G., "A 6b 1.1GSamples/s CMOS A/D converter," *ISSCC*, pp. 128–129, 2001.

[47] Genov, R. and Cauwenberghs, G., "Charge-mode parallel architecture for vector-matrix multiplication," *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 930–936, Oct. 2001.

[48] Gonzales, R. C. and Woods, R. E., *Digital Image Processing*. Delhi, India: Pearson Education Asia, 2002.

[49] Graupner, A., Schreiter, J., Getzlaff, S., and Schffny, R., "CMOS image sensor with mixed-signal processor erray," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 948–957, June 2003.

[50] Gruev, V. and Etienne-Cummings, R., "Implementation of steerable spatiotemporal image filters on the focal plane," *IEEE Transactions on Circuits and Systems II*, vol. 49, pp. 65–73, Apr. 2002.

[51] Harrison, R. R. and Koch, C., "An analog VLSI implementation of a visual interneuron: enhanced sensory processing through biophysical modeling," *International Journal of Neural Systems*, vol. 9, pp. 391–395, Oct. 1999.

[52] Harrison, R. R. and Koch, C., "A robust analog VLSI reichardt motion sensor," *Analog Integrated Circuits and Signal Processing*, vol. 24, pp. 213–229, Sept. 2000.

[53] Hasler, P., Bandyopadhyay, A., and Anderson, D. V., "High fill-factor imagers for neuromorphic processing enabled by floating gates," *EURASIP Journal on Applied Signal processing*, pp. 676–689, 2003.

[54] Hasler, P., Diorio, C., Minch, B. A., and Mead, C. A., *Advances in Neural Information Processing Systems 7*, ch. Single transistor learning synapses, pp. 817–824. Cambridge, MA: MIT Press, 1995.

[55] Hasler, P. and Lande, T. S., "Special issue on floating-gate devices, circuits, and systems," *IEEE Journal of Circuits and Systems*, vol. 48, Jan. 2001.

[56] Hasler, P. and Anderson, D. V., "Cooperative analog-digital signal processing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. IV, (Orlando, FL), pp. 3972–3975, May 2002.

[57] Hasler, P. and Bandyopadhyay, A., "A matrix transform imager and architecture," in *IEEE Sensors*, (Orlando), pp. 177–182, June 2002.

[58] Hasler, P., Bandyopadhyay, A., and Anderson, D. V., "Low-power analog image processing using transform imagers," in *IEEE Digital Signal Processing Workshop, 2002 and the 2nd Signal Processing Education Workshop*, pp. 333 – 338, Oct. 2002.

[59] Hasler, P., Bandyopadhyay, A., and Smith, P., "A matrix transform imager allowing high fill factor," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, (Phoenix, AZ), pp. 337–340, May 2002.

[60] Hasler, P. and Dugger, J., "Correlation learning rule in floating-gate pFET synapses," *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 65–73, Jan. 2001.

[61] Hasler, P. and Minch, B. A., *Floating-Gate Devices, Circuits, and Systems*. IEEE Press, 2002.

[62] Hasler, P., Minch, B. A., Dugger, J., and Diorio, C., "Adaptive circuits and synapses using pFET floating-gate devices," in *Learning in Silicon* (Cauwenbergs, G., ed.), pp. 33–65, Kluwer Academic, 1999.

[63] Hasler, P., Minch, B. A., Dugger, J., and Diorio, C., *Learning in Silicon*, ch. Adaptive Circuits and Synapses Using pFET Floating-Gate Devices, pp. 33–65. Kluwer Academic, 1999.

[64] Higgins, C. M. and Koch, C., "A modular multi-chip neuromorphic architecture for real-time visual motion processing," *Analog Integrated Circuits and Signal Processing*, vol. 24, pp. 195–211, Sept. 2000.

[65] Hyde, J., Humes, T., Diorio, C., Thomas, M., and Figueroa, M., "A 300-MS/s 14-bit digital-to-analog converter in logic CMOS," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 734–740, May 2003.

[66] Kandel, E. R., Schwartz, J. H., and Jessel, T. M., eds., *Principles of Neural Science*. New York: Elsevier, 1991.

[67] Kawahito, S., Yoshida, M., Sasaki, M., Umehara, K., Miyazaki, D., Tadokoro, Y., Murata, K., Doushou, S., and Matsuzawa, A., "A CMOS image sensor with analog two-dimensional DCT-based compression circuits for one-chip cameras," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 2030–2041, Dec. 1997.

[68] Khachab, N. and Ismail, M., "A 16 x 16 nonvolatile programmable analog vector-matrix multiplier," vol. 33, pp. 1502–1509, Oct. 1998.

[69] Kim, K. and Koh, J., "An area efficient DCT architecture for MPEG-2 video encoder," *IEEE Transactions on consumer electronics*, vol. 45, pp. 62–67, Feb. 1999.

[70] Kim, K., Lee, K., Jung, T., and Suh, K., "An 8-bit-resolution, 360-s write time nonvolatile analog memory based on differentially balanced constant-tunneling-current scheme (DBCS)," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 1758–1762, November 1998.

[71] Kinoshita, S., Morie, T., Nagata, M., and Iwata, A., "A PWM analog memory programming circuit for floating-gate MOSFETs with 75-μs programming time and 11-bit updating resolution," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 1286–1290, May 2003.

[72] Kordesch, A. and et. al., "A programming method for multilevel analog flash memory using coarse and fine sequence," *Int. Symposium on VLSI Technology, Systems, and Applications*, pp. 146–149, April 2001.

[73] Krenik, W. R., Hester, R. K., and DeGroat, R. D., "Current-mode flash A/D conversion based on current-splitting techniques," *Proceedings of the International Symposium on Circuits and Systems*, pp. 585 –588, 1992.

[74] Krymski, A. I., Bock, N. E., Tu, N., Blerkom, D. V., and Fossum, E. R., "A high-speed, 240-frames/s, 4.1-mpixel cmos sensor," *IEEE Transactions on Electron Devices*, vol. 50, pp. 130–135, Jan 2003.

[75] Kub, F., Moon, K., Mack, I., and Long, F., "Programmable analog vector-matrix multipliers," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 207–214, Feb. 1990.

[76] Kucic, M., Low, A., Hasler, P., and Neff, J., "A programmable continuous-time floating-gate fourier processor," *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 90–99, Jan. 2001.

[77] Kucic, M., Hasler, P., Dugger, J., and Anderson, D. V., "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *2001 Conference on Advanced Research in VLSI* (Brunvand, E. and Myers, C., eds.), pp. 148–162, IEEE Computer Society, March 2001.

[78] Kyomasu, M., "A new MOS imager using photodiode as current source," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 1116–1122, Aug. 1991.

[79] Lenzlinger, M. and Snow, E., "Fowler Nordheim tunneling into thermally grown sio," *Journal of Applied Physics*, vol. 40, no. 6, pp. 278–283, 1969.

[80] Liu, S., Kramer, J., Indiveri, G., Delbruck, T., and Douglas, R., *Analog VLSI: circuits and principles*. London, England: MIT press, 2002.

[81] Luo, Q. and Harris, J., "A novel integration of on-sensor wavelet compression for a CMOS imager," *Proceedings of the International Symposium on Circuits and Systems*, pp. 325–328, May 2002.

[82] Mahowald, M., "Analog VLSI chip for stereocorrespondence," in *Proceedings of the International Symposium on Circuits and Systems*, vol. 6, (London), pp. 347–350, May 1994.

[83] Mahowald, M., *An Analog VLSI Stereoscopic Vision System*. Boston, MA: Kluwer Academic Publishers, 1994.

[84] Mahowald, M. and Mead, C., "The silcon retina," *Scientific American*, vol. 264, no. 5, pp. 76–82, 1991.

[85] Martina, M., Molino, A., and Vacca, F., "Reconfigurable and low power 2D-DCT IP for ubiquitous multimedia streaming," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 26–29, 2002.

[86] Mead, C. A. and Ismail, M., eds., *Analog VLSI Implementation of Neural Systems*. Norwell, MA: Kluwer Academic Publishers, 1989.

[87] Mead, C., *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[88] Mead, C. A., "Neuromorphic electronic systems," *IEEE Proceedings*, vol. 78, pp. 1629–1636, Oct. 1990.

[89] Mehrvarz, H. and Kwok, C., "A novel multi-input floating-gate MOS four quadrant analog multiplier," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 1123–1131, Aug. 1996.

[90] Meitzler, R., Andreou, A., Strohbehn, K., and Jenkins, R., "A sampled–data motion chip," *Proc. Midwest Symposium on circuits and systems*.

[91] Moini, A., ed., *Vision Chips*. Boston/Dordrecht/London: Kluwer Academic Publishers, 1999.

[92] Morie, T., Fujita, O., and Uchimura, K., "Self learning analog neural network LSI with high-resolution nonvolatile analog memory and a partially serial weight-update architecture," *IEICE Trans. Electron*, vol. E80-C, no. 7, pp. 990–995, 1997.

[93] Paillet, F., Mercier, D., and Bernard, T., "Second generation programmable artificial retina," *Proc. IEEE ASIC/SOC Conf.*, pp. 304–309, September 1999.

[94] Park, J., Kwon, S., and Roy, K., "Low power reconfigurable DCT design based on sharing multiplication," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 3116–3119, 2002.

[95] Pereira, A., Brady, P., Bandyopadhyay, A., and Hasler, P., "Experimental investigations of floating–gate circuits for Delta-Sigma modulators," in *IEEE Midwest Circuits and Systems*, vol. 1, (Tulsa, OH), pp. 208–211, 2002.

[96] Rabaey, J. M., *Digital integrated circuits: A design perspective*. New Jersey 07458: Prentice hall, Uppr Saddle River, 1996.

[97] Razavi, B., *Design of analog CMOS integrated circuits*. New Delhi, India: TATA Mcgraw–Hill, 2002.

[98] Razavi, B. and Wooley, B. A., "A 12-b 5-msamples/s two–step CMOS A/D converter," *IEEE Journal of Solid-State Circuits*, pp. 1667–1678, 1992.

[99] Rolandi, P. L., Canegallo, R., Chioffi, E., Gerna, D., Issartel, C., Lhermet, F., Passotti, M., and Kramer, A., "A 1M-cell 6b/cell analog flash memory for digital storage," *IEEE Int. Solid-State Circuits Conference*, pp. 334–335, 1998.

[100] Sarpeshkar, R., Bair, W., and Koch, C., "Visual motion computation in analog VLSI using pulses," in *Advances in Neural Information Processing Systems 5* (Hanson, S., Cowan, J., and Giles, C., eds.), pp. 781–788, San Mateo, CA: Morgan Kaufman, 1993.

[101] Sjöström, U., Defilippis, I., Ansorge, M., and Pellandini, F., "Discrete cosine trsnsform chip for real–time viseo applications," *Proceedings of the International Symposium on Circuits and Systems*, pp. 1620–1623, May 1990.

[102] Smith, M. J. T. and Docef, A., *Digital Image Processing*. Riverdale, GA: Scientific Publishers, 1999.

[103] Song, H. and Kim, C., "An nMOS four-quadrant analog multiplier using simple two-input squaring circuits with source followers," *IEEE Journal of Solid-State Circuits*, vol. 25, pp. 841–848, June 1990.

[104] Spillman, L. and Werner, J. S., *Visual Perception: The Neurophysiological Foundations*. San Diego, CA: Academic Press, 1990.

[105] Spirig, T., Seitz, P., Vietze, O., and Heitger, F., "A smart CCD image sensor with real-time programmable parallel convolution capabilities," *IEEE Transactions on Circuits and Systems*, vol. 44, pp. 465–468, May 1997.

[106] Takayanagi, I., Shirakawa1, M., Mitani1, K., Sugawara, M., Iversen, S., Moholt, J., Nakamura, J., and Fossum, E. R., "11/4 inch 8.3m pixel digital output cmos aps for udtv application," *IEEE Int. Solid-State Circuits Conference*, pp. 216–217, 2003.

[107] Tam, S. and et. al., "Lucky electron model of channel hot-electron injection in MOSFETs," vol. 31, pp. 1116–1125, 1984.

[108] Tanner, J. and Mead, C., "An integrated analog optical motion sensor," in *VLSI Signal Processing II* (Brodersen, R. W. and Moscovitz, H. S., eds.), pp. 59–87, New York: IEEE, 1988.

[109] Tian, H., Fowler, B., and Gamal, A. E., "Analysis of temporal noise in CMOS photodiode active pixel sensor," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 92–101, Jan. 2001.

[110] Toumazou, C., Lidgey, F. J., and Haigh, D. G., *Analogue IC design: The current mode approach*. London: Peter Peregrinus, 1990.

[111] Traff, H., "Novel approach to high speed CMOS current comparators," *Electronics Letters*, pp. 310–312, 1992.

[112] Weckler, G. P., "Operation of p–n junction photodetectors in a photon flux integrating mode," *IEEE Journal of Solid-State Circuits*, pp. 65–73, 1967.

[113] Wong, L., Kwok, C., and Rigby, G., "A 1-v CMOS D/A converter with multi-input floating-gate MOSFET," *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 1386–1390, October 1999.

[114] Yadid-Pecht, O. and Fossum, E. R., "Wide intrascene dynamic range CMOS APS using dual sampling," *IEEE Transactions on Electron Devices*, vol. 44, pp. 1721–1723, Oct. 1997.

[115] Yadid-Pecht, O., Ginosar, R., and Diamand, Y. S., "A random access photodiode array for intelligent image capture," *IEEE Transactions on Electron Devices*, vol. 38, pp. 1772–1780, Aug. 1991.

[116] Yang, D. X. D., Fowler, B., and Gamal, A. E., "A nyquist rate pixel level adc for cmos image sensors," *Custom Integrated Circuits Conference*, pp. 237–240, May 1998.

[117] Yang, D. X. D., Gamal, A. E., Fowler, B., and Tian, H., "A 640x512 CMOS image sensor with ultrawide dynamic range floating–point pixel level ADC," *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 1821–1832, Dec 1999.

[118] Yee, C. and Buchwald, A., "A sampled-data switched-current analog 16-tap FIR filter with digitally programmable coefficients in 0.8μm CMOS," *IEEE Int. Solid-State Circuits Conference*, vol. 33, pp. 54–54, Feb 1997.

[119] Yuan, J. and Svensson, C., "High–speed CMOS circuit technique," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 62–70, Feb. 1989.